

uftrace 로 대규모 오픈소스 프로그램 및 리눅스 커널 쉽게 분석하기

KSC 2019 Tutorial

2019년 12월 18일

김홍규

honggyu.kp@gmail.com

uftrace 활용 시나리오 및 사례 정리

- 대표적으로 분석 가능한 대상 프로젝트들
 - GCC
 - Node.js
 - Clang/LLVM
 - Chromium
 - FFmpeg
 - etc.
- 다른 프로젝트들은 wiki 페이지 참조
 - <https://github.com/namhyung/uftrace/wiki>

튜토리얼 개요

- **uftrace 소개 및 기초 분석 활용**
 - uftrace 소개 및 사용법
 - C++ STL internal 분석 실습
- **uftrace 심화 분석 활용**
 - 오픈소스 프로젝트 분석
 - Clang/LLVM, Node.js, CPython 등
 - uftrace internal 소개
- **커널 및 시스템 프로그래밍 활용 (by 송태웅)**
 - 리눅스 커널 스토리지 스택 동작과정 분석 실습
 - VFS / File System / Block

uftrace 소개 및 기초 분석 활용

uftrace 소개

- C/C++ 프로그램에 대한 함수 호출 관계를 추적하는 도구
 - Creator: 김남형 <namhyung@kernel.org>
 - 리눅스 커널 개발자
 - 리눅스 내부 **perf** 성능 프로파일링 도구의 코드 리뷰어
 - linux/tools/perf/*
 - Linux perf 와 유사한 명령어 구조
 - 함수 실행 시 record 한 후 replay 와 같은 명령으로 분석

uftrace 소개

- uftrace 가 분석할 수 있는 것들
 - C/C++ 사용자(user-space) 함수
 - 컴파일 시 **-pg** 나 **-finstrument-functions** 옵션 필요
 - 또는 **-finstrument-functions-after-inlining** (clang only)
 - 또는 **-fxray-instrument** (clang only)
 - 라이브러리 함수 (library functions)
 - 리눅스 커널(kernel-space) 내부 함수
 - 시스템 이벤트들

대규모 프로젝트 분석의 어려움

- 다수의 개발자들이 사용하는 시간의 비율
 - 새 코드 작성(2%), 코드 수정(20%), **기존 코드 이해(78%)**
- 수많은 소스 코드들
 - 수많은 내부 자료구조, 다양한 알고리즘 및 패턴
- 각 프로젝트 별 개발 절차에 익숙하지 않음
 - 빌드 시스템, 내부 스크립트, 테스트 절차 등
- 핵심 구현에 추가된 최적화 및 수많은 예외 처리로 인한 코드 읽기의 어려움

uftrace

C/C++ 프로그램의 함수 실행 흐름을 추적하는 도구
A function (graph) tracer for C/C++ userspace programs

uftrace

<https://github.com/namhyung/uftrace>

Project Page

[namhyung / uftrace](#)

Code Issues Pull requests Actions Wiki Security Insights

Function (graph) tracer for user-space <https://uftrace.github.io/slide/>

trace function tracer

3,618 commits 22 branches 0 packages 16 releases 45 contributors GPL-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download ▾

Intuition and namhyung	utils: Fix memory leaks in tui	...	Latest commit c56514a 4 days ago
arch	arch/x86_64: Use proper size for mcount_loc array in bsearch()		28 days ago
check-deps	utils: Implement luajit script engine		last month
cmds	tui: Use --report option to start with report mode		6 days ago
doc	live: Reuse --record option to keep recorded data		6 days ago
gdb/uftrace	misc: Remove unused python variable		last year
libmcount	dynamic: Initialize a local variable properly		28 days ago
libtraceevent	build: Generate TRACEEVENT-CFLAGS in the objdir		11 months ago
misc	misc: Add raspbian to install-deps.sh		18 days ago
scripts	script: Add lua files to scripts directory		last month
tests	tests: Fix a minor difference in 062 arg_char test		13 days ago
utils	utils: Fix memory leaks in tui		14 hours ago

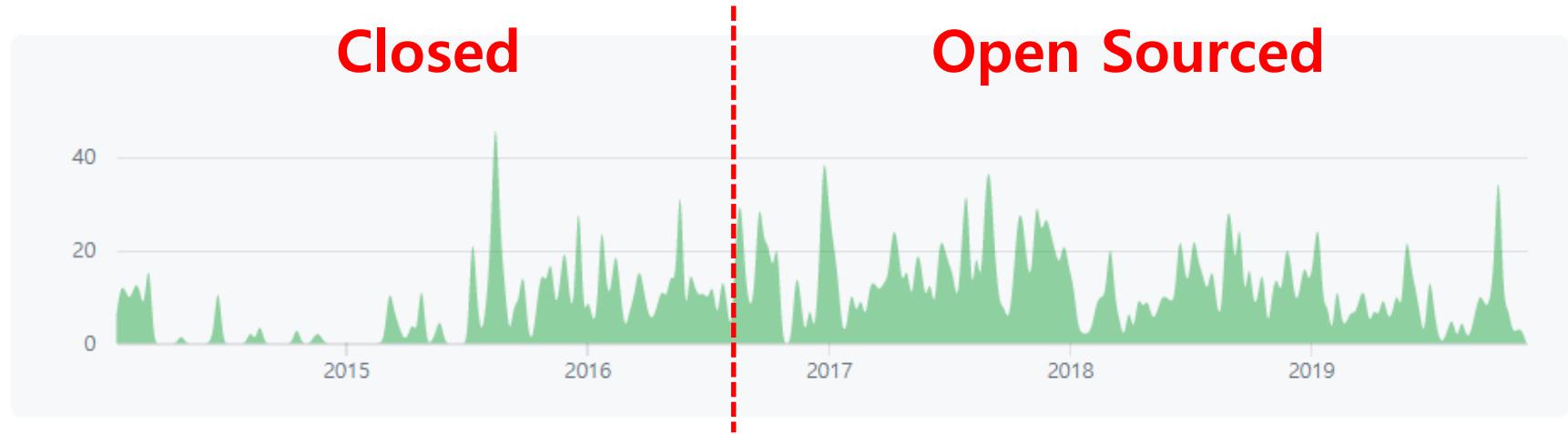
<https://github.com/namhyung/uftrace>

Project History

Jan 19, 2014 – Nov 26, 2019

Contributions: Commits ▾

Contributions to master, excluding merge commits



namhyung

#1

2,479 commits 118,406 ++ 48,398 --

A Google Trends chart titled "honggyukim" showing search interest over time. The y-axis represents search interest from 0 to 40, with horizontal grid lines at 0, 20, and 40. The x-axis shows years 2015 and 2018. The data is represented by an orange line with small peaks. The chart indicates a total of 329 commits, 25,685 ++, and 13,287 --.

honggyukim

#2

329 commits 25,685 ++ 13,287 --

40

20

2015 2018

\$

```
$ cat hello.c
```

```
$ cat hello.c
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

\$

```
$ gcc hello.c
```

```
$ gcc hello.c
```

```
$
```

```
$ gcc hello.c
```

```
$ ./a.out
```

```
$ gcc hello.c
```

```
$ ./a.out  
Hello, World!
```

```
$ gcc hello.c
```

```
$ ./a.out  
Hello, World!
```

이 프로그램은 내부에서
어떻게 실행이 되었을까?

```
$ ./node hello.js  
Hello, World!
```

이 프로그램은 내부에서
어떻게 실행이 되었을까?

```
$ ./clang hello.c
```



이 프로그램은 내부에서
어떻게 실행이 되었을까?

```
$ ./chrome
```



이 프로그램은 내부에서
어떻게 실행이 되었을까?

\$./chrome

{

Segmentation fault

```
$ ./chrome  
{
```

Segmentation fault (core dumped)

```
$ ./chrome
```



Segmentation fault (core dumped)

core dump 는 비정상 종료된
최종 상황만 저장하고 있음

```
$ gcc hello.c
```

```
$ gcc -pg hello.c
```

```
$ gcc -pg hello.c
```

```
$ ./a.out
```

```
$ gcc -pg hello.c
```

```
$ uftrace ./a.out
```

```
$ gcc -pg hello.c
```

```
$ utrace ./a.out
```

```
Hello, World!
```

```
$ gcc -pg hello.c
```

```
$ uctrace ./a.out
```

```
Hello, World!
```

#	DURATION	TID	FUNCTION
	1.447 us	[120218]	__monstartup();
	0.997 us	[120218]	__cxa_atexit();
		[120218]	main() {
	7.214 us	[120218]	printf();
	8.246 us	[120218]	} /* main */

```
$ gcc hello.c  
$ ./a.out
```

\$ gcc hello.c

\$./a.out

a.out

```
$ gcc hello.c
```

```
$ ./a.out
```

a.out



```
printf("Hello, World!")
```

```
$ gcc -pg hello.c
```

```
$ gcc -pg hello.c  
$ ./a.out
```

```
$ uftrace ./a.out
```

```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```

```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```

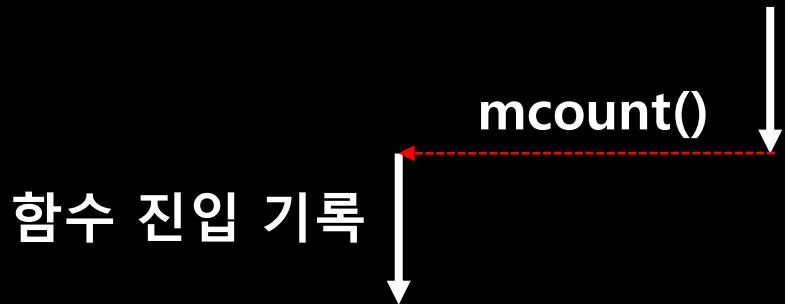


```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```

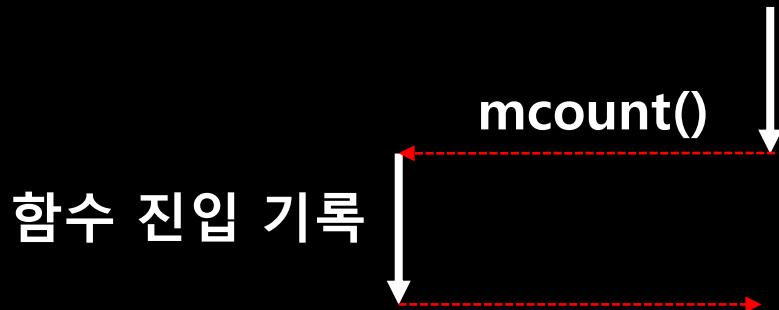
mcount()
↓



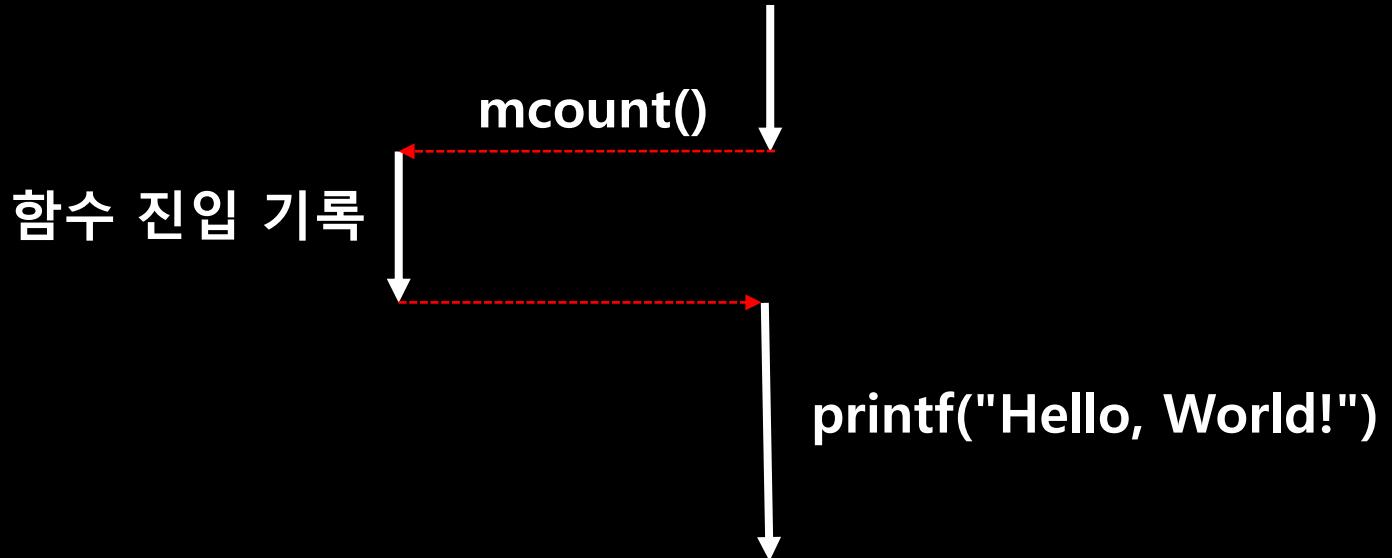
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



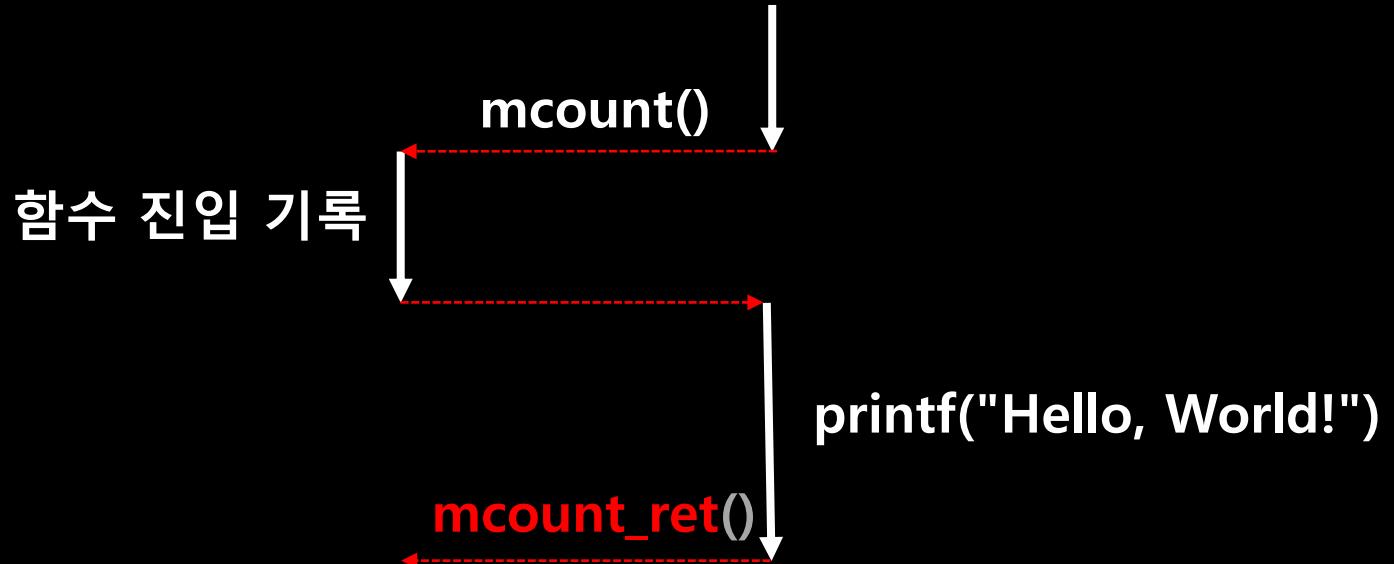
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



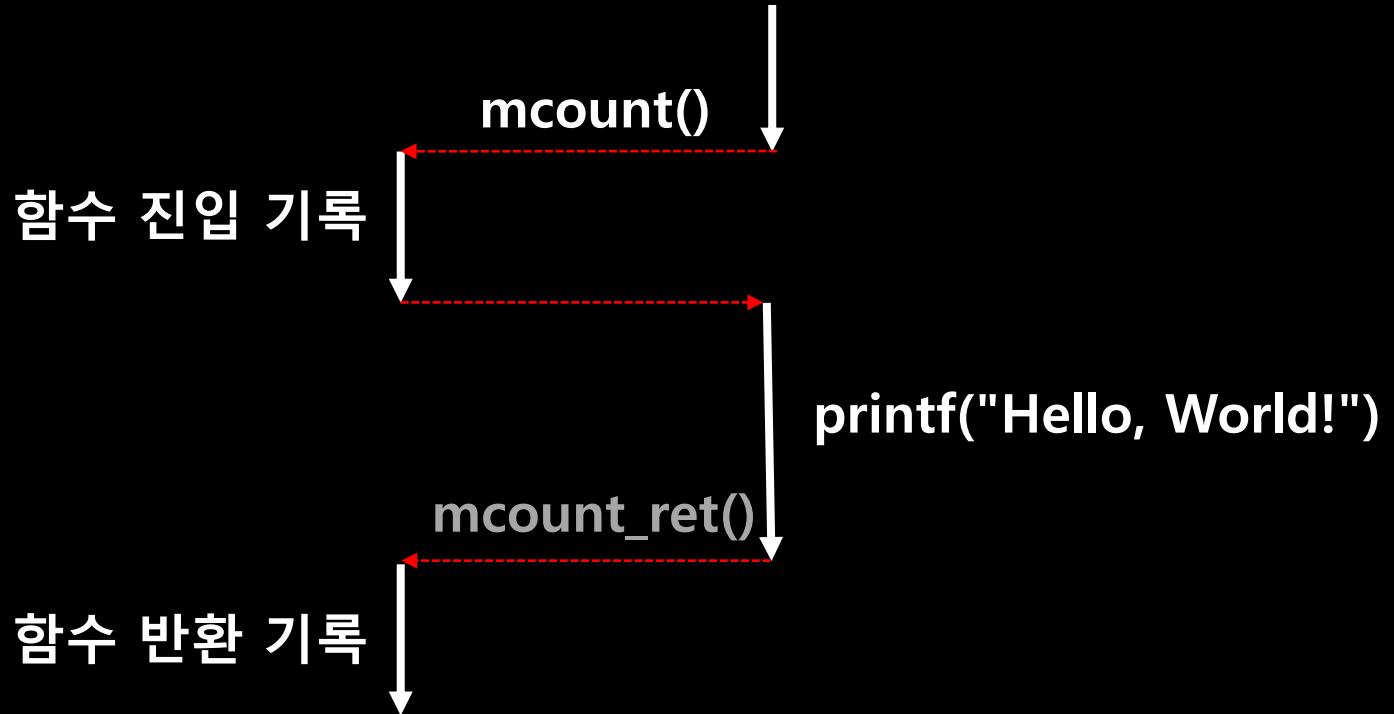
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



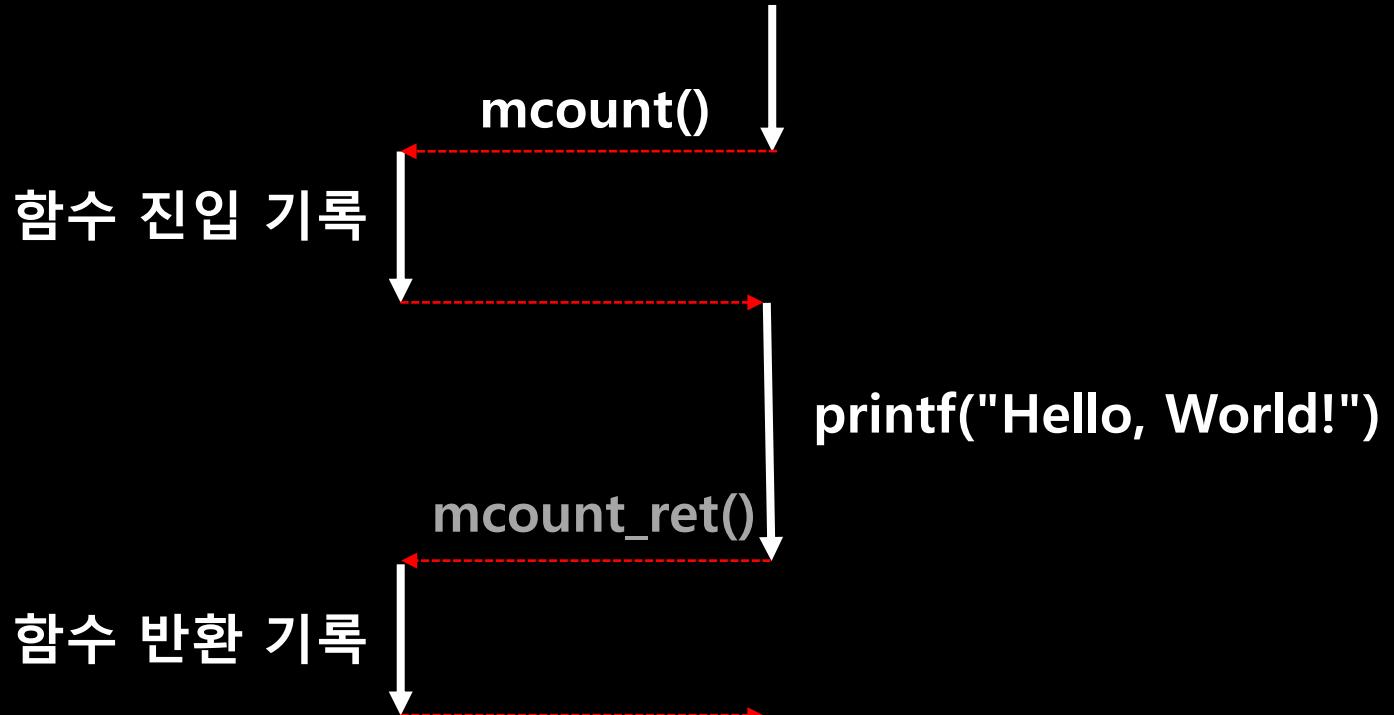
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



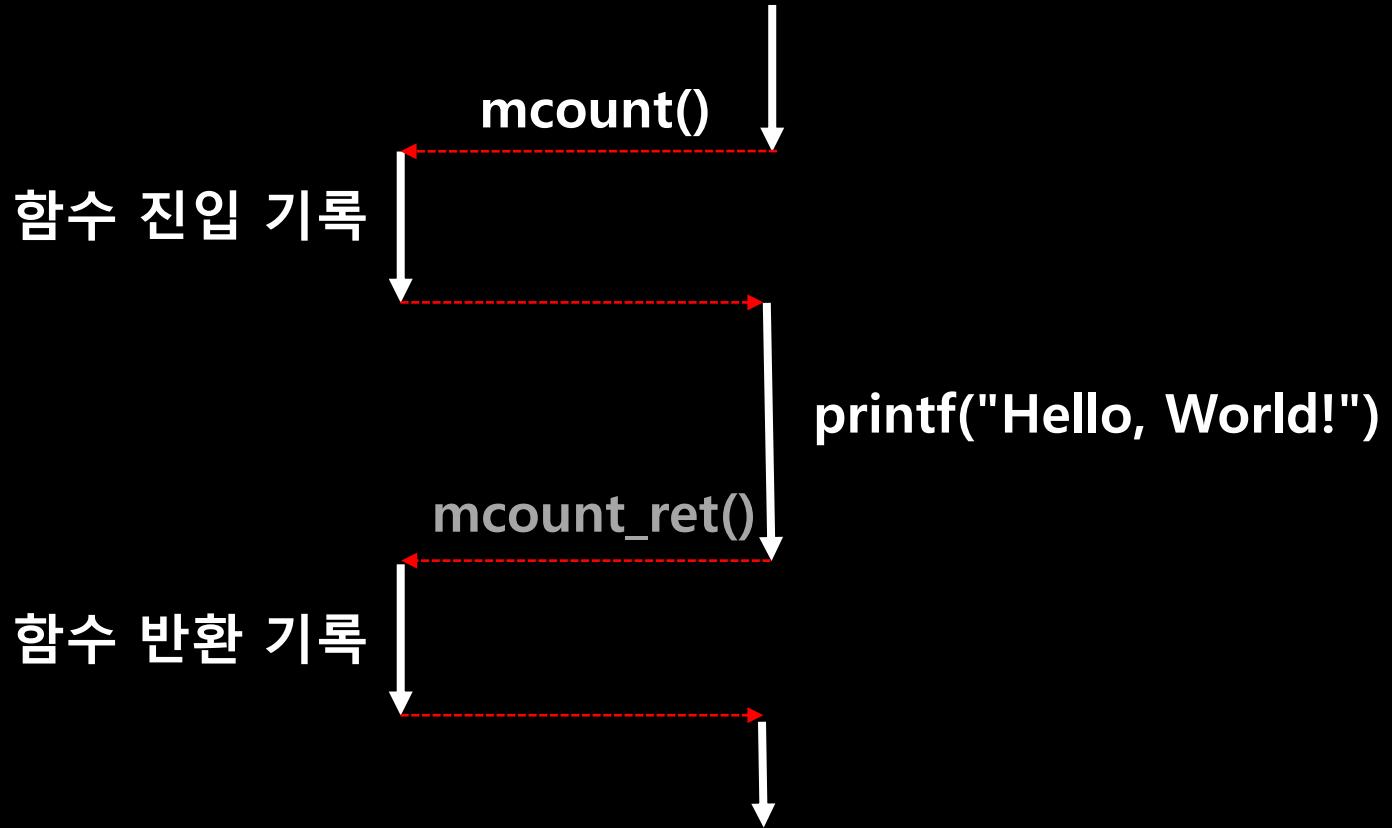
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



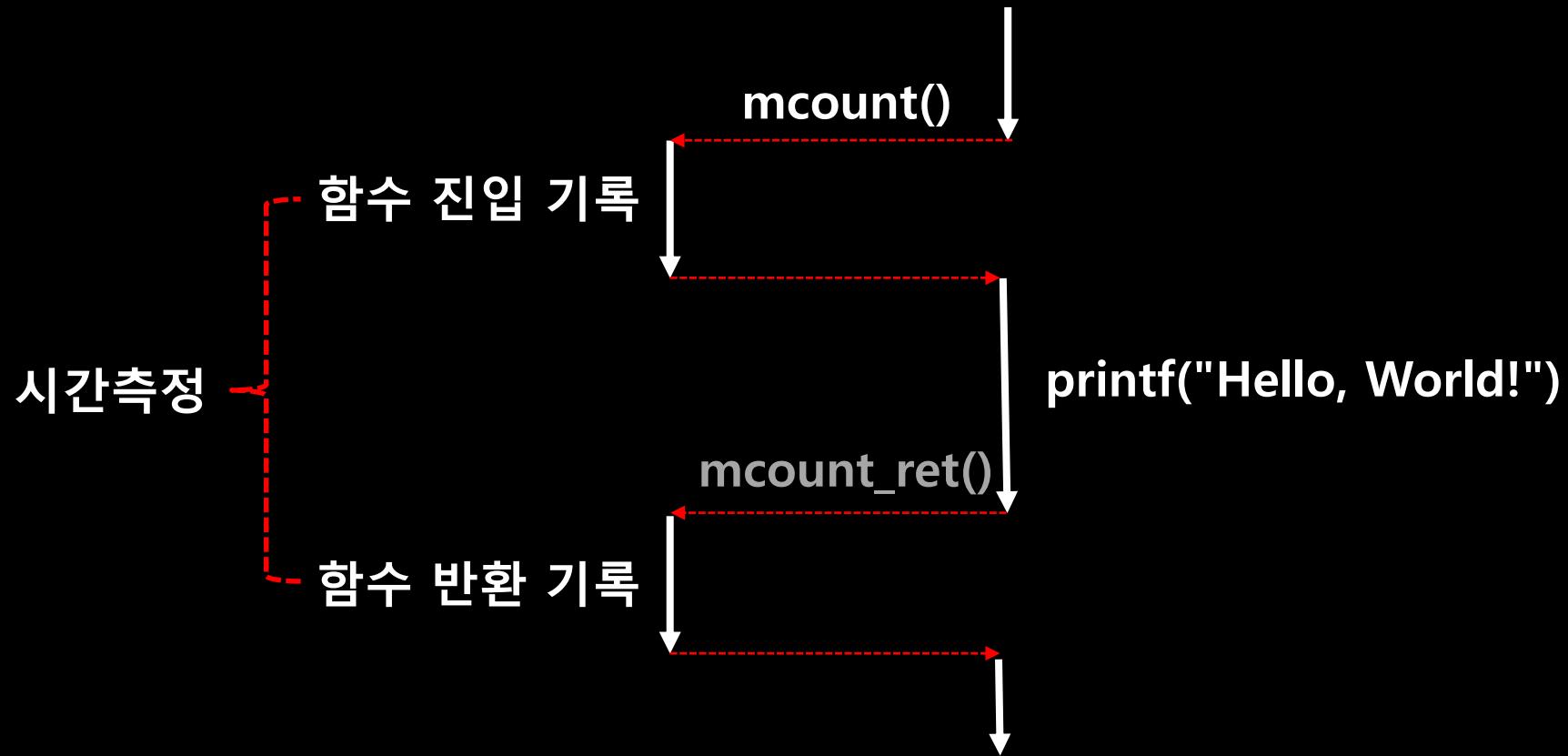
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



Quick Installation

```
$ git clone https://github.com/namhyung/uftrace && cd uftrace  
  
$ ./configure  
uftrace detected system features:  
...     prefix: /usr/local  
...     libelf: [ OFF ] - more flexible ELF data handling  
...     libdw: [ OFF ] - DWARF debug info support  
...     libpython2.7: [ OFF ] - python scripting support  
...     libncursesw: [ OFF ] - TUI support  
...     cxa_demangle: [ OFF ] - full demangler support with libstdc++  
...     perf_event: [ OFF ] - perf (PMU) event support  
...     schedule: [ OFF ] - scheduler event support  
...     capstone: [ OFF ] - full dynamic tracing support  
  
$ make  
$ sudo make install
```

모든 기능을 optional 하게 사용해 OFF 인 경우 해당 기능을 끄고 실행 가능

Quick Installation

```
$ git clone https://github.com/namhyung/uftrace && cd uftrace  
  
$ ./configure  
uftrace detected system features:  
...     prefix: /usr/local  
...     libelf: [ on ] - more flexible ELF data handling  
...     libdw: [ OFF ] - DWARF debug info support  
...     libpython2.7: [ OFF ] - python scripting support  
...     libncursesw: [ OFF ] - TUI support  
...     cxa_demangle: [ on ] - full demangler support with libstdc++  
...     perf_event: [ on ] - perf (PMU) event support  
...     schedule: [ on ] - scheduler event support  
...     capstone: [ OFF ] - full dynamic tracing support  
  
$ make  
$ sudo make install
```

일부 기능이 on 인경우 관련 라이브러리의 도움으로 더 많은 기능 사용 가능

Quick Installation

```
$ git clone https://github.com/namhyung/uftrace && cd uftrace  
  
# To install required packages  
$ sudo ./misc/install-deps.sh  
  
$ ./configure  
uftrace detected system features:  
...     prefix: /usr/local  
...     libelf: [ on ] - more flexible ELF data handling  
...     libdw: [ on ] - DWARF debug info support  
... libpython2.7: [ on ] - python scripting support  
... libncursesw: [ on ] - TUI support  
... cxa_demangle: [ on ] - full demangler support with libstdc++  
... perf_event: [ on ] - perf (PMU) event support  
... schedule: [ on ] - scheduler event support  
... capstone: [ on ] - full dynamic tracing support  
  
$ make  
$ sudo make install
```

./misc/install-deps.sh 스크립트로 관련 라이브러리들 전부 설치 가능

Quick Installation

```
$ git clone https://github.com/namhyung/uftrace && cd uftrace  
  
# To install required packages  
$ sudo ./misc/install-deps.sh  
  
$ ./configure --prefix=/home/honggyu/usr  
uftrace detected system features:  
...     prefix: /home/honggyu/usr  
...     libelf: [ on ] - more flexible ELF data handling  
...     libdw: [ on ] - DWARF debug info support  
...     libpython2.7: [ on ] - python scripting support  
...     libncursesw: [ on ] - TUI support  
...     cxa_demangle: [ on ] - full demangler support with libstdc++  
...     perf_event: [ on ] - perf (PMU) event support  
...     schedule: [ on ] - scheduler event support  
...     capstone: [ on ] - full dynamic tracing support  
  
$ make  
$ make install
```

--prefix로 설치 경로를 바꿀 수 있어서 root 권한이 없는 서버에도 설치 가능

uftrace 패키지 설치 (Ubuntu 18.04)

```
honggyu@honggyu-VirtualBox: ~
File Edit View Search Terminal Help
honggyu@honggyu-VirtualBox:~$ uftrace
Command 'uftrace' not found, but can be installed with:
sudo apt install uftrace

honggyu@honggyu-VirtualBox:~$ sudo apt install uftrace
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  uftrace
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 251 kB of archives.
After this operation, 1,022 kB of additional disk space will be used.
Get:1 http://kr.archive.ubuntu.com/ubuntu bionic/universe amd64 uftrace amd64 0.8.2-1 [251 kB]
Fetched 251 kB in 0s (3,777 kB/s)
Selecting previously unselected package uftrace.
(Reading database ... 139964 files and directories currently installed.)
Preparing to unpack .../uftrace_0.8.2-1_amd64.deb ...
Unpacking uftrace (0.8.2-1) ...
Setting up uftrace (0.8.2-1) ...
Processing triggers for man-db (2.8.3-2) ...
honggyu@honggyu-VirtualBox:~$ uftrace
Usage: uftrace [OPTION...]
                  [record|replay|live|report|info|dump|recv|graph|script]
                  [<program>]
Try `uftrace --help' or `uftrace --usage' for more information.
honggyu@honggyu-VirtualBox:~$
```

간단한 사용법

C/C++ (user) Function Tracing

- -pg 또는 다른 tracing 가능한 옵션으로 재컴파일 필요

```
void bar() {  
}  
void foo() {  
    bar();  
}  
int main() {  
    foo();  
    bar();  
}
```

C/C++ (user) Function Tracing

- -pg 또는 다른 tracing 가능한 옵션으로 재컴파일 필요

```
$ gcc foobar.c
```

```
void bar() {  
}  
void foo() {  
    bar();  
}  
int main() {  
    foo();  
    bar();  
}
```

C/C++ (user) Function Tracing

- -pg 또는 다른 tracing 가능한 옵션으로 재컴파일 필요

```
$ gcc foobar.c
```

```
<bar>:  
void bar() {  
    ret  
}  
void foo() {  
    bar();  
}  
int main() {  
    foo();  
    bar();  
}  
<foo>:  
call <bar>  
ret  
<main>:  
call <foo>  
call <bar>  
ret
```

C/C++ (user) Function Tracing

- -pg 또는 다른 tracing 가능한 옵션으로 재컴파일 필요

```
$ gcc -pg foobar.c
```

```
void bar() {                                <bar>:  
    call <mcount@plt>  
    ret  
  
}  
void foo() {                                <foo>:  
    bar();  
    call <mcount@plt>  
    call <bar>  
    ret  
  
int main() {                                <main>:  
    foo();  
    bar();  
    call <mcount@plt>  
    call <foo>  
    call <bar>  
    ret  
}
```

C/C++ (user) Function Tracing

- finstrument-functions 는 진입과 반환 모두 hooking

```
$ gcc -finstrument-functions foobar.c
```

```
void bar() {  
    <bar>:  
        call <_cyg_profile_func_enter@plt>  
        call <_cyg_profile_func_exit@plt>  
        ret  
}  
  
void foo() {  
    bar();  
    <foo>:  
        call <_cyg_profile_func_enter@plt>  
        call <bar>  
        call <_cyg_profile_func_exit@plt>  
        ret  
}  
  
int main() {  
    foo();  
    bar();  
    <main>:  
        call <_cyg_profile_func_enter@plt>  
        call <foo>  
        call <bar>  
        call <_cyg_profile_func_exit@plt>  
        ret  
}
```

C/C++ (user) Function Tracing

- `-fno-omit-frame-pointer`로 조금 더 안전한 코드 생성

```
$ gcc -finstrument-functions \
-fno-omit-frame-pointer foobar.c

void bar() {
}
void foo() {
    bar();
}
int main() {
    foo();
    bar();
}

<bar>:
    call <_cyg_profile_func_enter@plt>
    call <_cyg_profile_func_exit@plt>
    ret

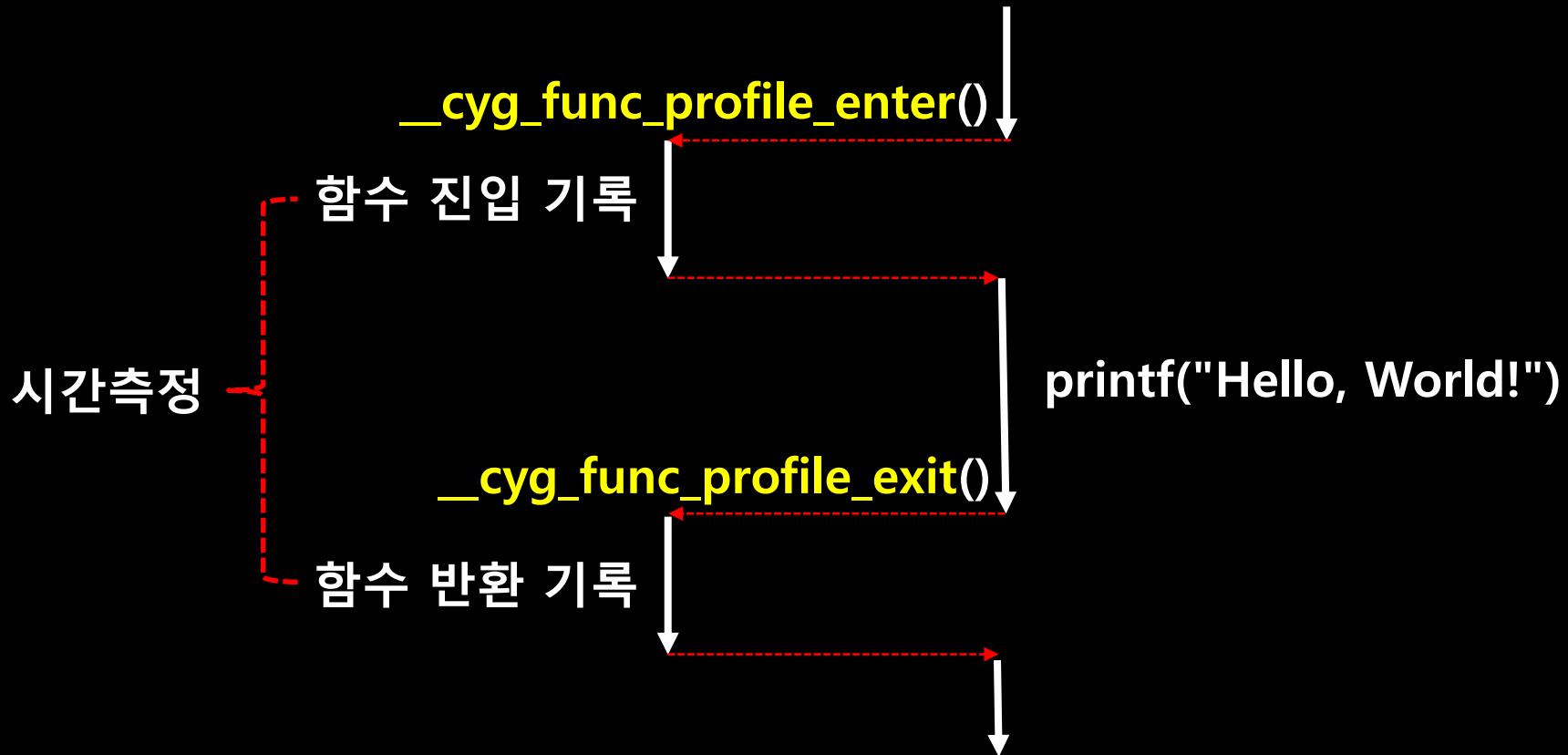
<foo>:
    call <_cyg_profile_func_enter@plt>
    call <bar>
    call <_cyg_profile_func_exit@plt>
    ret

<main>:
    call <_cyg_profile_func_enter@plt>
    call <foo>
    call <bar>
    call <_cyg_profile_func_exit@plt>
    ret
```

```
$ gcc -fprofile-instr-functions -fno-omit-frame-pointer hello.c
```

```
$ uftrace ./a.out
```

```
LD_PRELOAD=libmcount.so      a.out
```



C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c  
$ ./a.out
```

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c  
$ uftrace record a.out
```

uftrace record

- Run a command and record its trace data

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
$ uftrace record a.out
$ uftrace replay
# DURATION      TID      FUNCTION
  1.293 us  [11558] | __monstartup() ;
  0.814 us  [11558] | __cxa_atexit() ;
                     [11558] | main() {
                     [11558] |   foo() {
  0.156 us  [11558] |     bar();
  0.767 us  [11558] |   } /* foo */
  0.155 us  [11558] |   bar();
  1.140 us  [11558] | } /* main */
```

uftrace replay

- Print recorded function trace

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace live a.out
# DURATION      TID      FUNCTION
  1.293 us  [11558] | __monstartup();
  0.814 us  [11558] | __cxa_atexit();
                     [11558] | main() {
                     [11558] |   foo() {
  0.156 us  [11558] |     bar();
  0.767 us  [11558] |   } /* foo */
  0.155 us  [11558] |   bar();
  1.140 us  [11558] | } /* main */
```

uftrace live

- Trace functions in a command during live execution same as uftrace record and replay

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.293 us	[11558]	__monstartup();
	0.814 us	[11558]	__cxa_atexit();
		[11558]	main() {
		[11558]	foo() {
	0.156 us	[11558]	bar();
	0.767 us	[11558]	} /* foo */
	0.155 us	[11558]	bar();
	1.140 us	[11558]	} /* main */

uftrace (live)

- Trace functions in a command during live execution same as uftrace record and replay

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.293 us	[11558]	__monstartup();
	0.814 us	[11558]	__cxa_atexit();
		[11558]	main() {
		[11558]	foo() {
	0.156 us	[11558]	bar();
	0.767 us	[11558]	} /* foo */
	0.155 us	[11558]	bar();
	1.140 us	[11558]	} /* main */

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

# DURATION	TID	FUNCTION
1.293 us	[11558]	__monstartup();
0.814 us	[11558]	__cxa_atexit();
	[11558]	main() {
	[11558]	foo() {
0.156 us	[11558]	bar();
0.767 us	[11558]	} /* foo */
0.155 us	[11558]	bar();
1.140 us	[11558]	} /* main */

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

# DURATION	TID	FUNCTION
1.293 us	[11558]	__monstartup();
0.814 us	[11558]	__cxa_atexit();
	[11558]	main() {
	[11558]	foo() {
0.156 us	[11558]	bar();
0.767 us	[11558]	} /* foo */
0.155 us	[11558]	bar();
1.140 us	[11558]	} /* main */

C/C++ (user) Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.293 us	[11558]	__monstartup();
	0.814 us	[11558]	__cxa_atexit();
		[11558]	main() {
		[11558]	foo() {
	0.156 us	[11558]	bar();
	0.767 us	[11558]	} /* foo */
	0.155 us	[11558]	bar();
	1.140 us	[11558]	} /* main */

Library Function Tracing

- PLT hooking 을 통해 라이브러리 함수 호출 추적 가능

```
void bar() {  
    getpid();  
}  
void foo() {  
    bar();  
}  
int main() {  
    foo();  
    bar();  
}
```

Library Function Tracing

- PLT hooking 을 통해 라이브러리 함수 호출 추적 가능

```
$ gcc -pg foobar.c
```

```
void bar() {  
    getpid();  
}  
void foo() {  
    bar();  
}  
int main() {  
    foo();  
    bar();  
}
```

Library Function Tracing

- PLT hooking 을 통해 라이브러리 함수 호출 추적 가능

```
$ gcc -pg foobar.c
```

```
void bar() {  
    getpid();  
}  
  
void foo() {  
    bar();  
}  
  
int main() {  
    foo();  
    bar();  
}
```

```
<bar>:  
    call <mcount@plt>  
    call <getpid@plt> # indirect call in PLT  
    ret  
  
<foo>:  
    call <mcount@plt>  
    call <bar>  
    ret  
  
<main>:  
    call <mcount@plt>  
    call <foo>  
    call <bar>  
    ret
```

Library Function Tracing

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.100 us	[169246]	__monstartup();
	0.583 us	[169246]	__cxa_atexit();
		[169246]	main() {
		[169246]	foo() {
		[169246]	bar() {
	0.539 us	[169246]	getpid();
	1.192 us	[169246]	} /* bar */
	1.664 us	[169246]	} /* foo */
		[169246]	bar() {
	0.119 us	[169246]	getpid();
	0.499 us	[169246]	} /* bar */
	2.774 us	[169246]	} /* main */

Library Function Tracing

```
$ uftrace tests/t-fork
# DURATION      TID      FUNCTION
              [14528]  | main()  {
127.033 us  [14528]  |   fork();
              [14528]  |   wait()  {
              [14540]  |     } /* fork */
              [14540]  |     a()  {
              [14540]  |       b()  {
              [14540]  |         c()  {
1.507 us   [14540]  |           getpid();
2.987 us   [14540]  |         } /* c */
3.464 us   [14540]  |         } /* b */
3.854 us   [14540]  |         } /* a */
13.394 us  [14540]  |     } /* main */
799.270 us [14528]  |     } /* wait */
              [14528]  |     a()  {
              [14528]  |       b()  {
              [14528]  |         c()  {
2.410 us   [14528]  |           getpid();
3.470 us   [14528]  |         } /* c */
3.833 us   [14528]  |         } /* b */
4.144 us   [14528]  |         } /* a */
952.797 us [14528]  |     } /* main */
```

Nested Library Tracing

```
$ uftrace --nest-libcall --auto-args \
/usr/bin/clang hello.c
```

-l, --nest-libcall

Trace function calls between libraries.

By default, uftrace only record library call from
the main executable.

라이브러리 내부에서 다른 라이브러리를
호출하는 것도 tracing 해야 할 때 사용

Nested Library Tracing

```
$ uftrace -la \
/usr/bin/clang hello.c
```

-l, --nest-libcall

Trace function calls between libraries.

By default, uftrace only record library call from
the main executable.

라이브러리 내부에서 다른 라이브러리를
호출하는 것도 tracing 해야 할 때 사용

Nested Library Tracing

```
0.284 us [175968] | strlen("/usr/bin/ld") = 11;
[175968] | llvm::sys::commandLineFitsWithinSystemLimits() {
...
21.584 us [175968] | } /* llvm::sys::commandLineFitsWithinSystemLimits */
0.197 us [175968] | llvm::opt::ArgList::getLastArg();
0.420 us [175968] | memcpy(0x7ffc7ba7a020, 0x28a07d0, 384) = 0x7ffc7ba7a020;
0.323 us [175968] | strlen("/usr/lib/llvm-3.8/bin/clang") = 27;
[175968] | llvm::sys::ExecuteAndWait() {
0.360 us [175968] |     memcpy(0x7ffc7ba79b18, 0x2883dc0, 27) = 0x7ffc7ba79b18;
3.093 us [175968] |     access();
0.153 us [175968] |     std::__v2::system_category();
[175968] |     std::__cxx11::basic_string::__M_create() {
[175968] |         operator new() {
0.490 us [175968] |             malloc(28) = 0x28a1150;
1.053 us [175968] |         } /* operator new */
1.566 us [175968] |     } /* std::__cxx11::basic_string::__M_create */
0.253 us [175968] |     memcpy(0x28a1150, 0x2883dc0, 27) = 0x28a1150;
247.286 us [175968] |     posix_spawn();
[175968] |     operator delete() {
0.590 us [175968] |         free(0x28a1150);
1.500 us [175968] |     } /* operator delete */
[175968] |     waitpid(175980, 0x7ffc7ba79bfc, 0) {
...
...
```

재컴파일 없이 시스템에 배포된 clang 이미지에 적용해도
일부 라이브러리 함수 호출 확인 가능

Linux Kernel Function Tracing

```
$ gcc -pg hello.c
```

Linux Kernel Function Tracing

```
$ gcc -pg hello.c
```

```
$ sudo uftrace -k a.out  
Hello, World!
```

Linux Kernel Function Tracing

```
$ gcc -pg hello.c
```

```
$ sudo uftrace -k a.out
```

```
Hello, World!
```

#	DURATION	TID	FUNCTION	리눅스 커널 내부 함수들
	0.395 us	[8926]	__monstartup();	
	0.354 us	[8926]	__cxa_atexit();	
		[8926]	main() {	
		[8926]	printf() {	
	0.572 us	[8926]	sys_newfstat();	
	1.316 us	[8926]	__do_page_fault();	
	4.123 us	[8926]	} /* puts */	
		[8926]	fflush() {	리눅스 커널
	5.229 us	[8926]	sys_write();	내부 함수들
	6.454 us	[8926]	} /* fflush */	
	11.171 us	[8926]	} /* main */	

Event Tracing (sched event)

```
$ uftrace t-fork
# DURATION      TID      FUNCTION
              [14983] |  main()  {
225.620 us  [14983] |    fork();
              [14983] |    wait()  {
              [14983] |      /* linux:sched-out */
              [14995] |    } /* fork */
              [14995] |    a()  {
              [14995] |      b()  {
              [14995] |        c()  {
1.033 us   [14995] |          getpid();
2.280 us   [14995] |        } /* c */
2.677 us   [14995] |        } /* b */
3.020 us   [14995] |        } /* a */
11.131 us  [14995] |    } /* main */
676.988 us  [14983] |      /* linux:sched-in */
695.312 us  [14983] |    } /* wait */
              [14983] |    a()  {
              [14983] |      b()  {
              [14983] |        c()  {
2.067 us   [14983] |          getpid();
3.067 us   [14983] |        } /* c */
3.444 us   [14983] |        } /* b */
3.841 us   [14983] |        } /* a */
950.334 us  [14983] |    } /* main */
```

Event Tracing (sched event)

```
$ uftrace --no-event t-fork
# DURATION      TID      FUNCTION
#           [14983] |  main()  {
225.620 us [14983] |    fork();
#           [14983] |    wait()  {
#           [14995] |          } /* fork */
#           [14995] |          a()  {
#           [14995] |            b()  {
#           [14995] |              c()  {
1.033 us  [14995] |                  getpid();
2.280 us  [14995] |                  } /* c */
2.677 us  [14995] |                  } /* b */
3.020 us  [14995] |                  } /* a */
11.131 us  [14995] |          } /* main */

695.312 us  [14983] |          } /* wait */
#           [14983] |          a()  {
#           [14983] |            b()  {
#           [14983] |              c()  {
2.067 us  [14983] |                  getpid();
3.067 us  [14983] |                  } /* c */
3.444 us  [14983] |                  } /* b */
3.841 us  [14983] |                  } /* a */
950.334 us  [14983] |          } /* main */
```

PMU: Performance Monitoring Unit

```
$ uftrace -T main@read=pmu-cycle t-abc
```

PMU: Performance Monitoring Unit

```
$ uctrace record -T main@read=pmu-cycle t-abc

$ uctrace replay -f duration
# DURATION      FUNCTION
  1.466 us | __monstartup();
  1.127 us | __cxa_atexit();
             | main() {
             | /* read:pmu-cycle
             |     (cycle=158792, instructions=89990) */
             |     a() {
             |     | b() {
             |     |     c() {
  1.050 us |         getpid();
  2.786 us |     } /* c */
  3.447 us | } /* b */
  4.003 us | } /* a */
             | /* diff:pmu-cycle
             |     (cycle=+6119, instructions=+5641, IPC=0.92) */
  9.520 us | } /* main */
```

PMU: Performance Monitoring Unit

```
$ uctrace record -T main@read=pmu-cycle t-abc
```

```
$ uctrace replay -f duration
```

```
# DURATION      FUNCTION
```

```
1.466 us | __monstartup();  
1.127 us | __cxa_atexit();  
| main() {  
|   /* read:pmu-cycle  
|     (cycle=158792, instructions=89990) */  
|   a() {  
|     b() {  
|       c() {  
1.050 us |         getpid();  
2.786 us |       } /* c */  
3.447 us |     } /* b */  
4.003 us |   } /* a */  
|   /* diff:pmu-cycle  
|     (cycle=+6119, instructions=+5641, IPC=0.92) */  
9.520 us | } /* main */
```

함수 진입부터 반환 시점까지 진행된
cycle 수와 명령어 개수 정보

다른 read trigger 이벤트들

`-T <func>@read=<event>`

`$ man utrace record`

...

The **read trigger** is to read some information at runtime. The result will be recorded as (builtin) events at the beginning and the end of a given function. As of now, following **events** are supported:

- "proc/statm": process memory stat from /proc filesystem
- "page-fault": number of page faults using getrusage(2)
- "pmu-cycle": cpu cycles and instructions using Linux perf-event syscall
- "pmu-cache": (cpu) cache-references and misses using Linux perf-event syscall
- "pmu-branch": branch instructions and misses using Linux perf-event syscall

Event Recording

- `perf_event_paranoid` 값에 의해 읽지 못하는 event 를 읽을 수 없는 경우
 - 아래와 같이 값이 3인 경우 권한 없음

```
$ cat /proc/sys/kernel/perf_event_paranoid  
3
```

- 값을 아래와 같이 1로 설정해야 함

```
$ echo 1 | sudo tee /proc/sys/kernel/perf_event_paranoid  
1
```

user + lib + kernel 통합 tracing

- 여러 단계의 프로그램 실행 흐름 분석 가능

```
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, World! \n";
}

$ g++ -pg hello.cpp
```

user + lib + kernel 통합 tracing

- 여러 단계의 프로그램 실행 흐름 분석 가능
 - user program

```
$      uftrace -f +module -F main a.out  
Hello, World!
```

user + lib + kernel 통합 tracing

- 여러 단계의 프로그램 실행 흐름 분석 가능
 - user program

```
$      uftrace -f +module -F main a.out
Hello, World!
# DURATION      TID      MODULE NAME      FUNCTION
[ 66406]          a.out | main()  {
[ 66406]          a.out |   std::operator<<()  {
```

```
69.884 us [ 66406]          a.out |   } /* std::operator<< */
71.131 us [ 66406]          a.out | } /* main */
```

user + lib + kernel 통합 tracing

- 여러 단계의 프로그램 실행 흐름 분석 가능
 - user program -> **libstdc++.so** -> **libc.so**

```
$      uftrace -f +module -F main --nest-libcall a.out
Hello, World!
# DURATION      TID      MODULE NAME      FUNCTION
              [ 66406]      a.out | main()  {
              [ 66406]      a.out |     std::operator<<()  {

1.300 us [ 66406] libstdc++.so.6.0 |     strlen();
              [ 66406] libstdc++.so.6.0 |     std::__ostream_insert()  {
0.940 us [ 66406] libstdc++.so.6.0 |         std::basic_ostream::sentry::sentry();
              [ 66406] libstdc++.so.6.0 |         fwrite()  {

37.150 us [ 66406] libstdc++.so.6.0 |             } /* fwrite */
              [ 66406] libstdc++.so.6.0 |             std::basic_ostream::sentry::~sentry();
42.223 us [ 66406] libstdc++.so.6.0 |                 } /* std::__ostream_insert */
69.884 us [ 66406]      a.out |             } /* std::operator<< */
71.131 us [ 66406]      a.out |         } /* main */
```

user + lib + kernel 통합 tracing

- 여러 단계의 프로그램 실행 흐름 분석 가능
 - user program -> **libstdc++.so** -> **libc.so** -> **kernel**

```
$ sudo uftrace -f +module -F main --nest-libcall --kernel a.out
Hello, World!
# DURATION      TID      MODULE NAME      FUNCTION
              [ 66406]      a.out | main() {
              [ 66406]      a.out |     std::operator<<() {
19.820 us  [ 66406]  [kernel] |       __do_page_fault();
1.300 us   [ 66406] libstdc++.so.6.0 |       strlen();
              [ 66406] libstdc++.so.6.0 |       std::__ostream_insert() {
0.940 us   [ 66406] libstdc++.so.6.0 |           std::basic_ostream::sentry::sentry();
              [ 66406] libstdc++.so.6.0 |           fwrite() {
6.894 us   [ 66406]  [kernel] |               sys_newfstat();
18.750 us  [ 66406]  [kernel] |               sys_write();
37.150 us  [ 66406] libstdc++.so.6.0 |           } /* fwrite */
0.923 us   [ 66406] libstdc++.so.6.0 |           std::basic_ostream::sentry::~sentry();
42.223 us  [ 66406] libstdc++.so.6.0 |           } /* std::__ostream_insert */
69.884 us  [ 66406]      a.out |       } /* std::operator<< */
71.131 us  [ 66406]      a.out |   } /* main */
```

Filters

불필요한 함수가 너무 많을 때 사용하는 필터들

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

```
$ gcc -pg foobar.c
```

```
$ uftrace -D 2 a.out
# DURATION           TID      FUNCTION
  1.207 us [ 2699] | __monstartup();
  0.627 us [ 2699] | __cxa_atexit();
                     [ 2699] | main() {
                     [ 2699] |   foo() {
  0.141 us [ 2699] |     bar();
  0.784 us [ 2699] |   } /* foo */
  0.091 us [ 2699] |   bar();
  1.496 us [ 2699] | } /* main */
```

-D DEPTH, **--depth=DEPTH**

Set global trace limit in nesting level.

```
$ gcc -pg foobar.c
```

```
$ uftrace -D 2 a.out
```

#	DURATION	TID	FUNCTION
	1.965 us	[2724]	__monstartup();
	0.638 us	[2724]	__cxa_atexit();
		[2724]	main() {
	0.405 us	[2724]	foo();
	0.109 us	[2724]	bar();
	1.478 us	[2724]	} /* main */

-D DEPTH, **--depth=DEPTH**

Set global trace limit in nesting level.

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

```
$ gcc -pg foobar.c
```

```
$ uftrace -F foo a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

-F FUNC, --filter=FUNC

Set filter to trace selected functions only.

```
$ gcc -pg foobar.c
```

```
$ uftrace -F foo a.out
```

#	DURATION	TID	FUNCTION
		[2762]	foo() {
0.259 us		[2762]	bar();
2.253 us		[2762]	} /* main */

-F FUNC, --filter=FUNC

Set filter to trace selected functions only.

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

```
$ gcc -pg foobar.c
```

```
$ uftrace -N foo a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

-N FUNC, --notrace=FUNC

Set filter not to trace selected functions
(and children)

```
$ gcc -pg foobar.c
```

```
$ uftrace -N foo a.out
```

#	DURATION	TID	FUNCTION
	1.947 us	[2801]	__monstartup();
	0.790 us	[2801]	__cxa_atexit();
		[2801]	main() {
	0.144 us	[2801]	bar();
	1.617 us	[2801]	} /* main */

-N FUNC, --notrace=FUNC

Set filter not to trace selected functions
(and children)

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

```
$ gcc -pg foobar.c
```

```
$ uftrace -C foo a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

-C FUNC, --caller-filter=FUNC

Set filter to trace **callers** of selected functions only.

```
$ gcc -pg foobar.c
```

```
$ uftrace -C foo a.out
```

#	DURATION	TID	FUNCTION
		[2848]	main() {
0.599	us	[2848]	foo();
2.654	us	[2848]	} /* main */

-C FUNC, --caller-filter=FUNC

Set filter to trace callers of selected functions only.

```
$ gcc -pg foobar.c
```

```
$ uftrace a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

```
$ gcc -pg foobar.c
```

```
$ uftrace -t 200ns a.out
```

#	DURATION	TID	FUNCTION
	1.207 us	[2699]	__monstartup();
	0.627 us	[2699]	__cxa_atexit();
		[2699]	main() {
		[2699]	foo() {
	0.141 us	[2699]	bar();
	0.784 us	[2699]	} /* foo */
	0.091 us	[2699]	bar();
	1.496 us	[2699]	} /* main */

-t TIME, --time-filter=TIME

Do not show small functions under the
time threshold.

```
$ gcc -pg foobar.c
```

```
$ uftrace -t 200ns a.out
```

#	DURATION	TID	FUNCTION
	1.357 us	[2862]	__monstartup();
	0.602 us	[2862]	__cxa_atexit();
		[2862]	main() {
	0.423 us	[2862]	foo();
	1.073 us	[2862]	} /* main */

-t TIME, --time-filter=TIME

Do not show small functions under the
time threshold.

Report

시간 순서가 아닌 전체 결과에 대한 통합 결과

```
$ gcc -pg foobar.c
```

```
$ gcc -pg foobar.c  
$ utrace record a.out
```

```
$ gcc -pg foobar.c  
$ uftrace record a.out  
$ uftrace report
```

uftrace report

- Print statistics and summary for trace data

```
$ gcc -pg foobar.c  
$ uftrace record a.out  
$ uftrace report
```

Total time	Self time	Calls	Function
10.500 us	10.500 us	1	_monstartup
7.900 us	3.400 us	1	main
3.500 us	2.400 us	1	foo
2.100 us	2.100 us	2	bar
1.600 us	1.600 us	1	_cxa_atexit

uftrace report

- Print statistics and summary for trace data

```
$ gcc -pg foobar.c  
$ uftrace record a.out  
$ uftrace report -s total
```

Total time	Self time	Calls	Function
10.500 us	10.500 us	1	_monstartup
7.900 us	3.400 us	1	main
3.500 us	2.400 us	1	foo
2.100 us	2.100 us	2	bar
1.600 us	1.600 us	1	_cxa_atexit

uftrace report

- Print statistics and summary for trace data

```
$ gcc -pg foobar.c  
$ uftrace record a.out  
$ uftrace report -s self
```

Total time	Self time	Calls	Function
10.500 us	10.500 us	1	<code>_monstartup</code>
7.900 us	3.400 us	1	<code>main</code>
3.500 us	2.400 us	1	<code>foo</code>
2.100 us	2.100 us	2	<code>bar</code>
1.600 us	1.600 us	1	<code>__cxa_atexit</code>

uftrace report

- Print statistics and summary for trace data

```
$ gcc -pg foobar.c
$ uftrace record a.out
$ uftrace report -s call
```

Total time	Self time	Calls	Function
2.100 us	2.100 us	2	bar
1.600 us	1.600 us	1	<u>__cxa_atexit</u>
10.500 us	10.500 us	1	<u>__monstartup</u>
3.500 us	2.400 us	1	foo
7.900 us	3.400 us	1	main

uftrace report

- Print statistics and summary for trace data

Recording Function Arguments and Return Values

함수 인자값과 반환값을 함께 기록하는 방법

```
$ gcc -pg fibonacci.c
```

```
$ gcc -pg fibonacci.c  
$ uftrace ./a.out 5  
fib(5) = 5
```

```
$ gcc -pg fibonacci.c
$ uctrace ./a.out 5
fib(5) = 5
# DURATION      TID      FUNCTION
  0.620 us [31321] | __monstartup();
  0.456 us [31321] | __cxa_atexit();
                     [31321] | main() {
  1.478 us [31321] |     atoi();
                     [31321] |     fib() {
                     [31321] |         fib() {
                     [31321] |             fib() {
  0.155 us [31321] |                 fib();
  0.123 us [31321] |                 fib();
  0.883 us [31321] |             } /* fib */
  0.125 us [31321] |             fib();
  1.483 us [31321] |             } /* fib */
                     [31321] |             fib() {
  0.125 us [31321] |                 fib();
  0.125 us [31321] |                 fib();
  0.774 us [31321] |             } /* fib */
  2.716 us [31321] |             } /* fib */
  4.382 us [31321] |             printf();
  9.456 us [31321] |         } /* main */
```

```
$ gcc -pg fibonacci.c
$ uftrace -A fib@arg1 ./a.out 5
fib(5) = 5
# DURATION      TID      FUNCTION
0.770 us [31365] | __monstartup();
0.492 us [31365] | __cxa_atexit();
[31365] | main() {
1.507 us [31365] |     atoi();
[31365] |     fib(5) {
[31365] |         fib(4) {
[31365] |             fib(3) {
1.293 us [31365] |                 fib(2);
0.172 us [31365] |                 fib(1);
2.295 us [31365] |             } /* fib */
0.157 us [31365] |             fib(2);
3.025 us [31365] |             } /* fib */
[31365] |             fib(3) {
0.150 us [31365] |                 fib(2);
0.155 us [31365] |                 fib(1);
0.917 us [31365] |             } /* fib */
5.232 us [31365] |         } /* fib */
4.856 us [31365] |         printf();
12.697 us [31365] |     } /* main */
```

```
$ gcc -pg fibonacci.c
$ uctrace -A fib@arg1 -R fib@retval ./a.out 5
fib(5) = 5
# DURATION      TID      FUNCTION
  0.718 us [31379] | __monstartup();
  0.464 us [31379] | __cxa_atexit();
[31379] | main() {
  1.442 us [31379] |     atoi();
[31379] |     fib(5) {
[31379] |         fib(4) {
[31379] |             fib(3) {
  1.395 us [31379] |                 fib(2) = 1;
  0.174 us [31379] |                 fib(1) = 1;
  2.562 us [31379] |             } = 2; /* fib */
  0.157 us [31379] |             fib(2) = 1;
  3.330 us [31379] |             } = 3; /* fib */
[31379] |             fib(3) {
  0.152 us [31379] |                 fib(2) = 1;
  0.154 us [31379] |                 fib(1) = 1;
  0.959 us [31379] |             } = 2; /* fib */
  5.351 us [31379] |         } = 5; /* fib */
  5.729 us [31379] |         printf();
13.627 us [31379] |     } /* main */
```

```
$ uftrace -A fib@arg1 -R fib@retval ./a.out 5
```

ARGUMENTS

```
<argument>      := <symbol> "@" <specs>
<specs>        := <spec> | <spec> "," <spec>
<spec>          := ( <int_spec> | <float_spec> | <ret_spec> )
<int_spec>     := "arg" N [ "/" <format> [ <size> ] ] [ "%" ( <reg> | <stack> ) ]
<float_spec>   := "fparg" N [ "/" ( <size> | "80" ) ] [ "%" ( <reg> | <stack> ) ]
<ret_spec>     := "retval" [ "/" <format> [ <size> ] ]
<format>        := "i" | "u" | "x" | "s" | "c" | "f" | "S"
<size>          := "8" | "16" | "32" | "64"
<reg>           := <arch-specific register name> # "rdi", "xmm0", "r0", ...
<stack>         := "stack" [ "+" ] <offset>
```

```
$ uftrace -A fib@arg1 -R fib@retval ./a.out 5
```

ARGUMENTS

```
<argument>      := <symbol> "@" <specs>
<specs>         := <spec> | <spec> "," <spec>
<spec>          := ( <int_spec> | <float_spec> | <ret_spec> )
<int_spec>       := "arg" N [ "/" <format> [ <size> ] ] [ "%" ( <reg> | <stack> ) ]
<float_spec>     := "fparg" N [ "/" ( <size> | "80" ) ] [ "%" ( <reg> | <stack> ) ]
<ret_spec>        := "retval" [ "/" <format> [ <size> ] ]
<format>         := "i" | "u" | "x" | "s" | "c" | "f" | "S"
<size>          := "8" | "16" | "32" | "64"
<reg>            := <arch-specific register name> # "rdi", "xmm0", "r0", ...
<stack>          := "stack" [ "+" ] <offset>
```

```
$ uftrace -A fib@arg1 -R fib@retval ./a.out 5
```

ARGUMENTS

```
<argument>      := <symbol> "@" <specs>
<specs>         := <spec> | <spec> "," <spec>
<spec>          := ( <int_spec> | <float_spec> | <ret_spec> )
<int_spec>       := "arg" N [ "/" <format> [ <size> ] ] [ "%" ( <reg> | <stack> ) ]
<float_spec>     := "fparg" N [ "/" ( <size> | "80" ) ] [ "%" ( <reg> | <stack> ) ]
<ret_spec>        := "retval" [ "/" <format> [ <size> ] ]
<format>         := "i" | "u" | "x" | "s" | "c" | "f" | "S"
<size>          := "8" | "16" | "32" | "64"
<reg>            := <arch-specific register name> # "rdi", "xmm0", "r0", ...
<stack>          := "stack" [ "+" ] <offset>
```

```
$ uftrace -A fib@arg1 -R fib@retval ./a.out 5
```

ARGUMENTS

```
<argument>      := <symbol> "@" <specs>
<specs>        := <spec> | <spec> "," <spec>
<spec>          := ( <int_spec> | <float_spec> | <ret_spec> )
<int_spec>     := "arg" N [ "/" <format> [ <size> ] ] [ "%" ( <reg> | <stack> ) ]
<float_spec>   := "fparg" N [ "/" ( <size> | "80" ) ] [ "%" ( <reg> | <stack> ) ]
<ret_spec>     := "retval" [ "/" <format> [ <size> ] ]
<format>        := "i" | "u" | "x" | "s" | "c" | "f" | "S"
<size>          := "8" | "16" | "32" | "64"
<reg>           := <arch-specific register name> # "rdi", "xmm0", "r0", ...
<stack>         := "stack" [ "+" ] <offset>
```

uftrace with DWARF (a standardized debugging data format)

함수 인자와 반환 타입 자동 인식

-a / --auto-args option

```
$ gcc -pg -g fibonacci.c
$ uftrace ./a.out 5
fib(5) = 5
# DURATION      TID      FUNCTION
  0.620 us [31321] | __monstartup();
  0.456 us [31321] | __cxa_atexit();
                     [31321] | main() {
  1.478 us [31321] |     atoi();
                     [31321] |     fib() {
                     [31321] |         fib() {
                     [31321] |             fib() {
  0.155 us [31321] |                 fib();
  0.123 us [31321] |                 fib();
  0.883 us [31321] |             } /* fib */
  0.125 us [31321] |             fib();
  1.483 us [31321] |             } /* fib */
                     [31321] |             fib() {
  0.125 us [31321] |                 fib();
  0.125 us [31321] |                 fib();
  0.774 us [31321] |             } /* fib */
  2.716 us [31321] |             } /* fib */
  4.382 us [31321] |             printf();
  9.456 us [31321] |         } /* main */
```

```
$ gcc -pg -g fibonacci.c
$ uftrace -a ./a.out 5
fib(5) = 5
# DURATION      TID      FUNCTION
  0.718 us [31379] | __monstartup();
  0.464 us [31379] | __cxa_atexit();
                     [31379] | main(2, 0x7ffc8dc59d98) {
  1.442 us [31379] |     atoi();
                     [31379] |     fib(5) {
                     [31379] |         fib(4) {
                     [31379] |             fib(3) {
  1.395 us [31379] |                 fib(2) = 1;
  0.174 us [31379] |                 fib(1) = 1;
  2.562 us [31379] |             } = 2; /* fib */
  0.157 us [31379] |             fib(2) = 1;
  3.330 us [31379] |             } = 3; /* fib */
                     [31379] |             fib(3) {
  0.152 us [31379] |                 fib(2) = 1;
  0.154 us [31379] |                 fib(1) = 1;
  0.959 us [31379] |             } = 2; /* fib */
  5.351 us [31379] |         } = 5; /* fib */
  5.729 us [31379] |         printf("%d\n") = 2;
13.627 us [31379] |     } = 0; /* main */
```

Visualization

uftrace dump

--chrome / --flame-graph

```
$ gcc -pg fibonacci.c
```

```
$ gcc -pg fibonacci.c
$ uftrace record ./a.out 5
fib(5) = 5
```

```
$ gcc -pg fibonacci.c
$ uftrace record ./a.out 5
fib(5) = 5
$ uftrace dump
```

uftrace dump

- Print raw tracing data in the data files

```
$ gcc -pg fibonacci.c
$ uftrace record ./a.out 5
fib(5) = 5
$ uftrace dump --chrome
```

--chrome

Show JSON style output as used by the Google Chrome tracing facility.

```
$ gcc -pg fibonacci.c
$ uftrace record ./a.out 5
fib(5) = 5
$ uftrace dump --chrome
```

```
{"traceEvents": [
{"ts":5913706403443,"ph":"B","pid":32256,"name":"__monstartup"},  

 {"ts":5913706403444,"ph":"E","pid":32256,"name":"__monstartup"},  

 {"ts":5913706403447,"ph":"B","pid":32256,"name":"__cxa_atexit"},  

 {"ts":5913706403447,"ph":"E","pid":32256,"name":"__cxa_atexit"},  

 {"ts":5913706403448,"ph":"B","pid":32256,"name":"main"},  

 {"ts":5913706403448,"ph":"B","pid":32256,"name": "atoi"},  

 {"ts":5913706403450,"ph":"E","pid":32256,"name": "atoi"},  

 {"ts":5913706403450,"ph":"B","pid":32256,"name": "fib"},  

 {"ts":5913706403450,"ph":"B","pid":32256,"name": "fib"},  

 ...
 {"ts":5913706403452,"ph":"E","pid":32256,"name": "fib"},  

 {"ts":5913706403453,"ph":"E","pid":32256,"name": "fib"},  

 {"ts":5913706403453,"ph":"E","pid":32256,"name": "fib"},  

 {"ts":5913706403453,"ph":"B","pid":32256,"name": "printf"},  

 {"ts":5913706403457,"ph":"E","pid":32256,"name": "printf"},  

 {"ts":5913706403458,"ph":"E","pid":32256,"name": "main"}  

], "metadata": {  

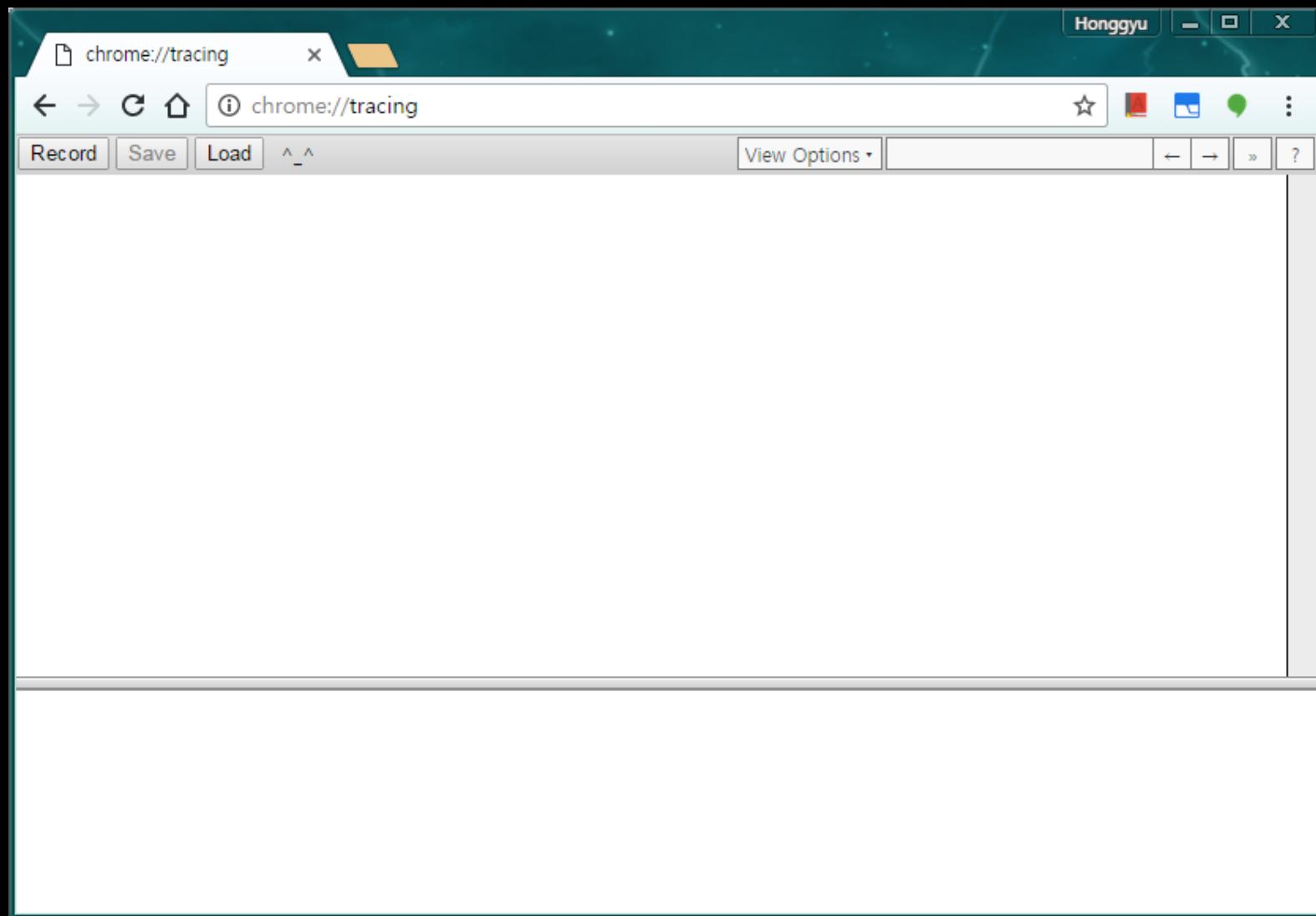
 "command_line": "uftrace record fibonacci 5 ",  

 "recorded_time": "Thu Sep 22 22:31:17 2016"  

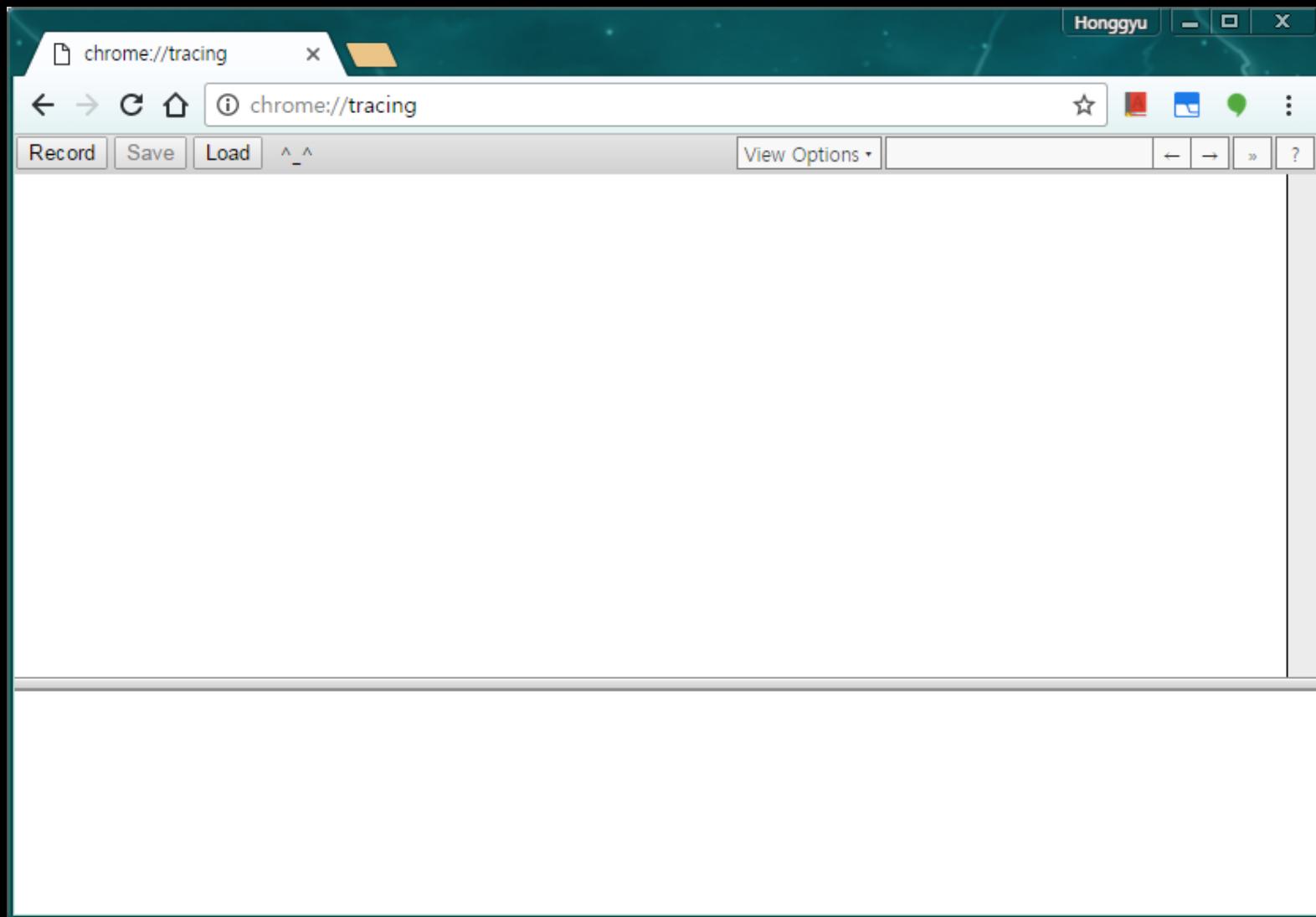
} }
```

```
$ gcc -pg fibonacci.c
$ uftrace record ./a.out 5
fib(5) = 5
$ uftrace dump --chrome > fib.json
```

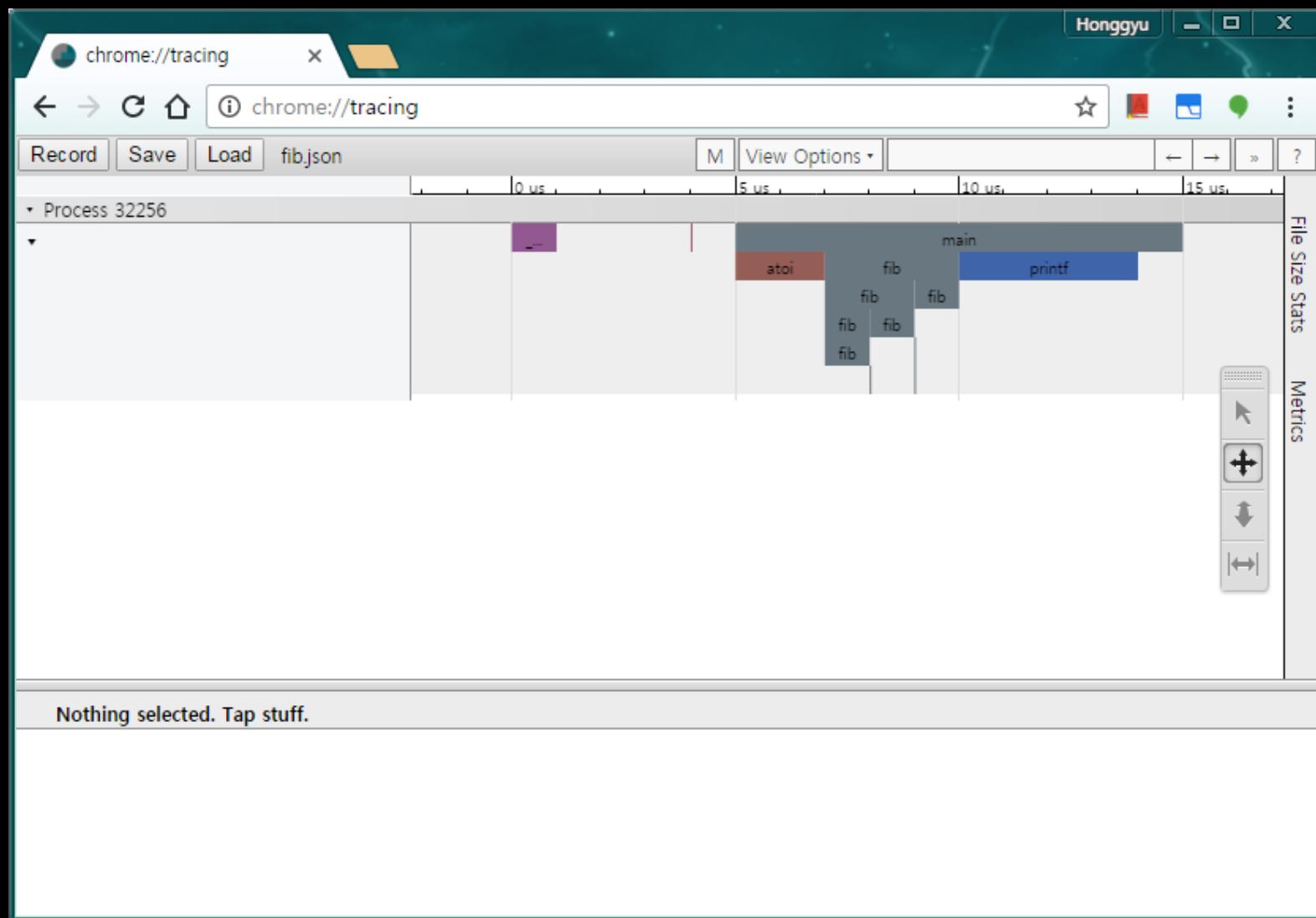
1. Open Chrome Browser



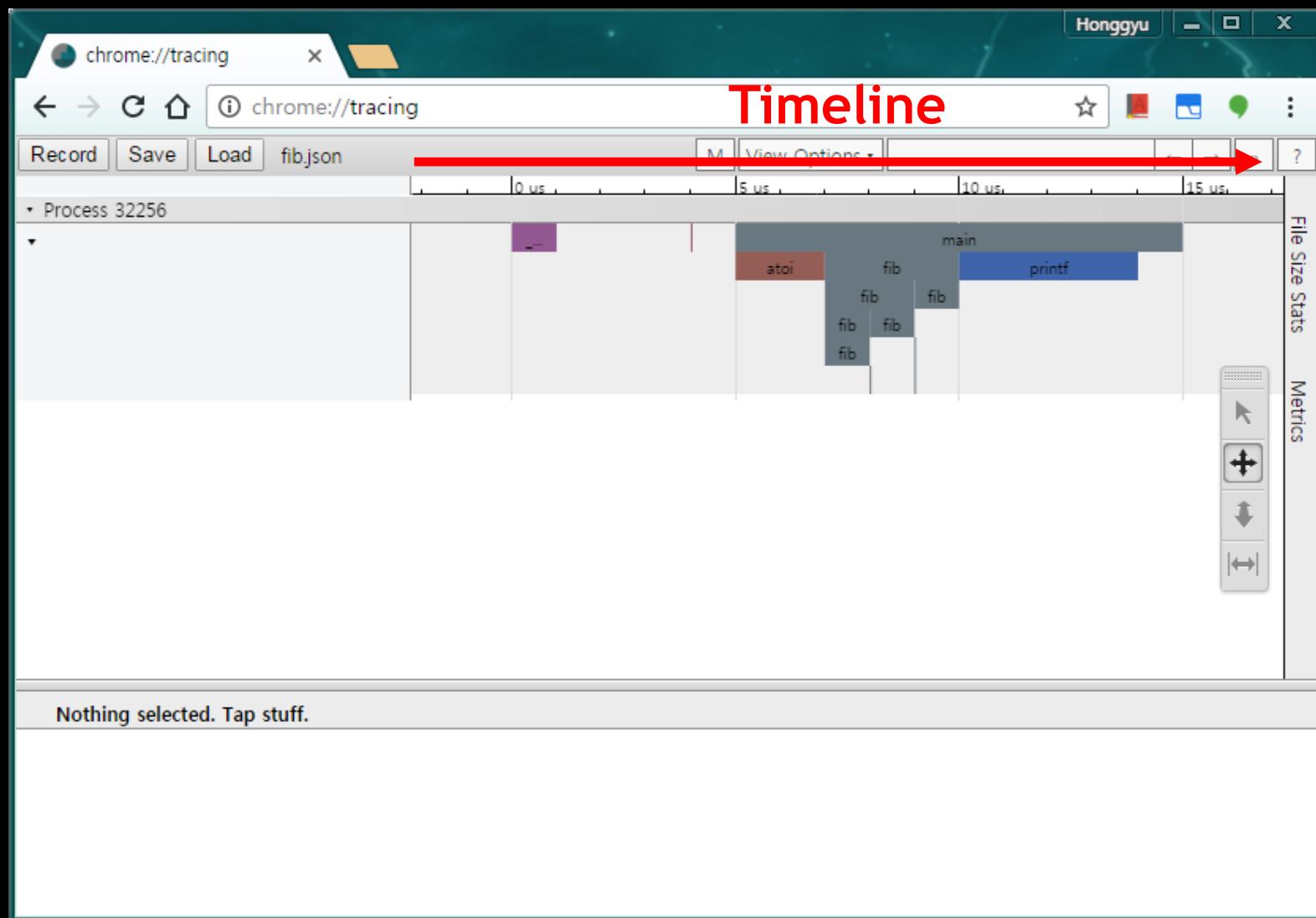
1. Open Chrome Browser
2. Load JSON file in **chrome://tracing**



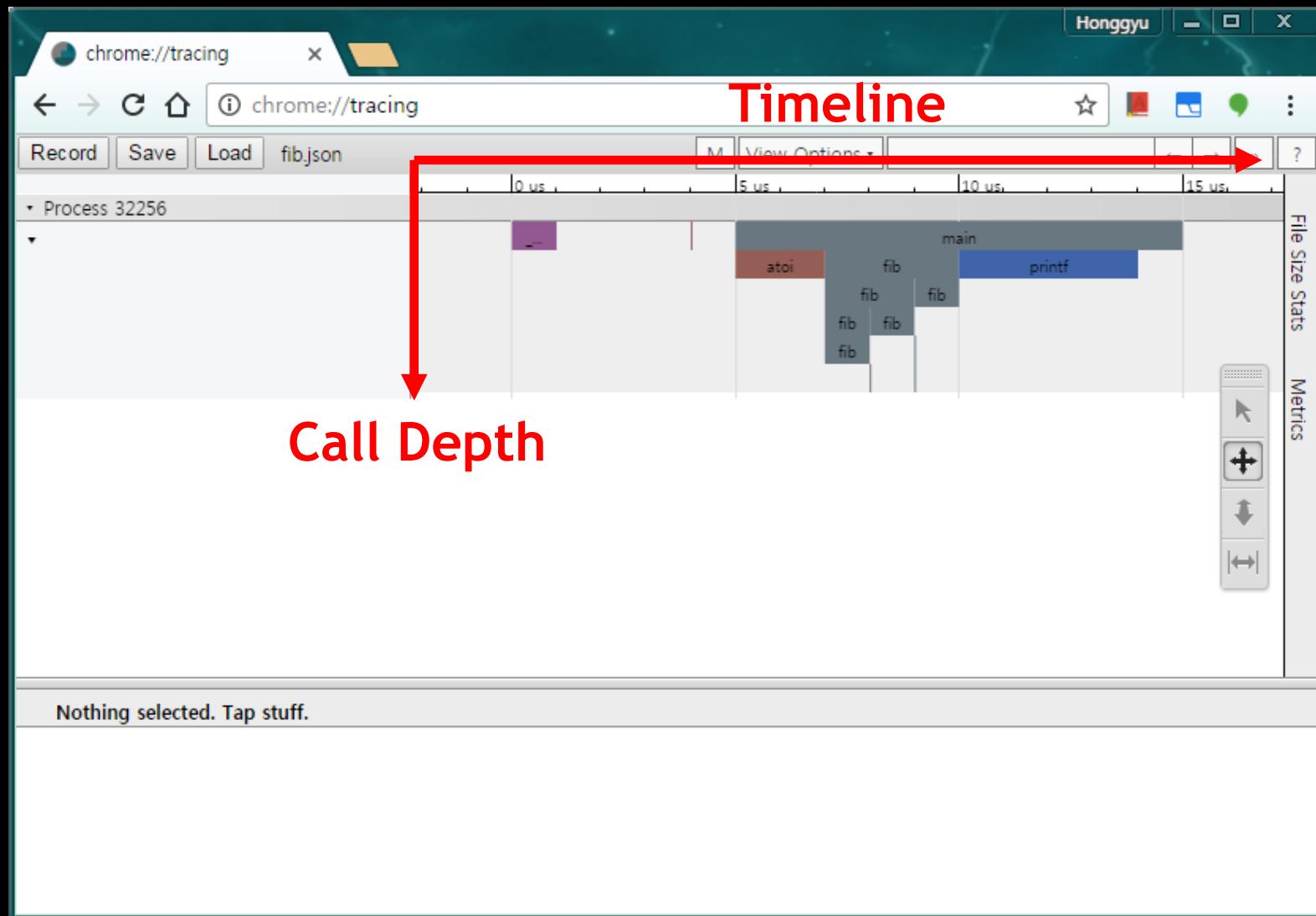
1. Open Chrome Browser
2. Load JSON file in **chrome://tracing**



1. Open Chrome Browser
2. Load JSON file in **chrome://tracing**



1. Open Chrome Browser
2. Load JSON file in `chrome://tracing`



chrome://tracing

Chrome | chrome://tracing

Record Save Load ^_^ View Options

Hongyu

Chrome Tracing Help

Navigation

- w/s Zoom in/out (+shift: faster)
- a/d Pan left/right (+shift: faster)
- /shift-TAB Select previous event
- ←/TAB Select next event

Mouse Controls

- click Select event
- alt-mousewheel Zoom in/out

 **Select mode**

- drag Box select
- ctrl-click/drag Add events to the current selection
- double click Select all events with same title

 **Pan mode**

- drag Pan the view

 **Zoom mode**

- drag Zoom in/out by dragging up/down

 **Timing mode**

- drag Create or move markers
- double click Set marker range to slice

General

- 1-4 Switch mouse mode
- shift Hold for temporary select
- space Hold for temporary pan
- / Search
- enter Step through search results
- f Zoom into selection
- z/0 Reset zoom and pan
- g/G Toggle 60hz grid
- ∨ Highlight VSync
- h Toggle low/high details
- m Mark current selection
- P Select power samples over current selection interval
- ` Show or hide the scripting console
- ? Show help

JSON 파일을 HTML로 변환

- "trace2html"으로 .json 파일을 .html 파일로 변환 가능.
 - <https://github.com/catapult-project/catapult/blob/master/tracing/bin/trace2html>

```
$ trace2html trace-fib.json
```

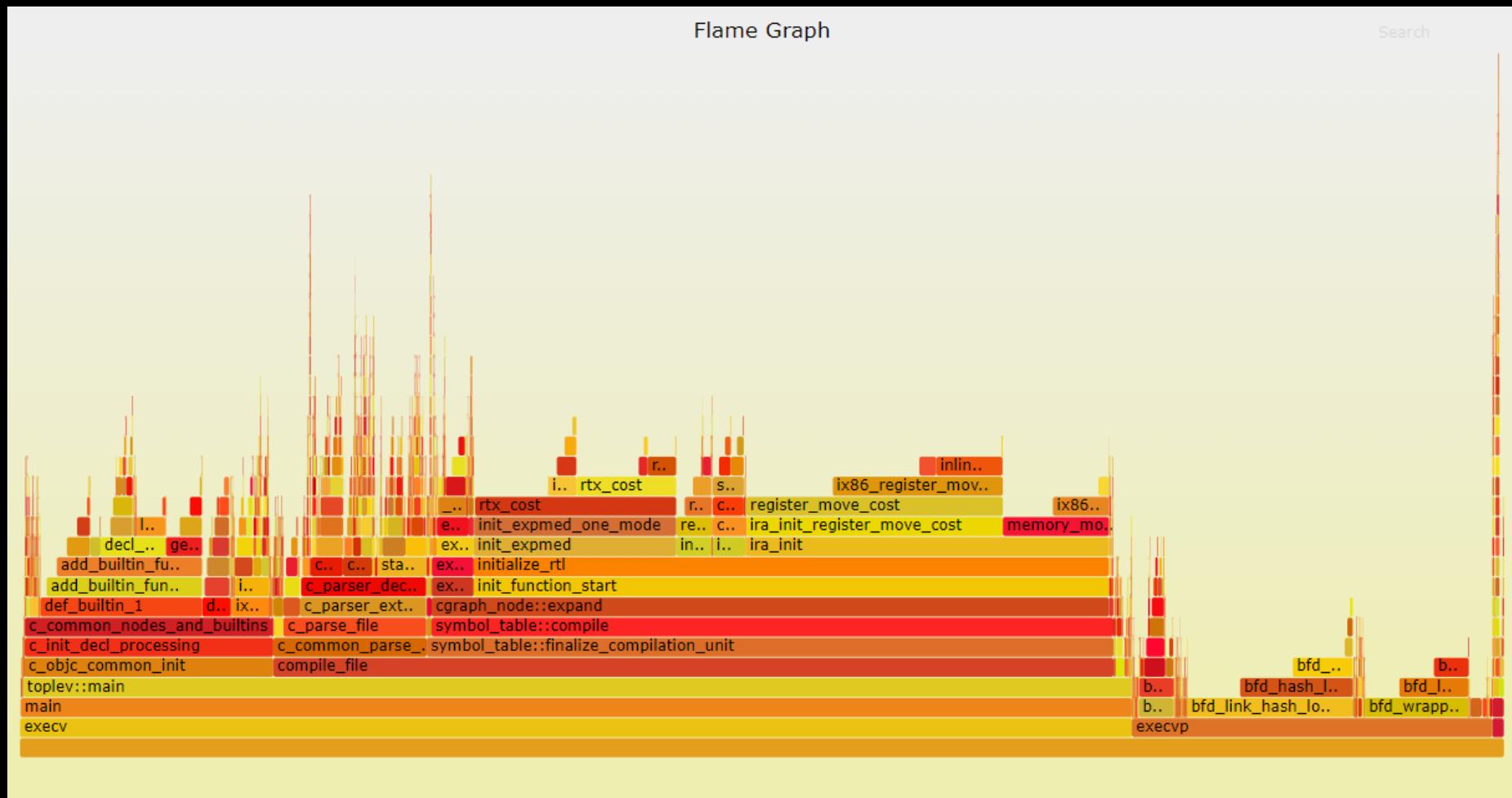
JSON 파일을 HTML로 변환

- "trace2html"으로 .json 파일을 .html 파일로 변환 가능.
 - <https://github.com/catapult-project/catapult/blob/master/tracing/bin/trace2html>

```
$ trace2html trace-fib.json  
trace-fib.html
```

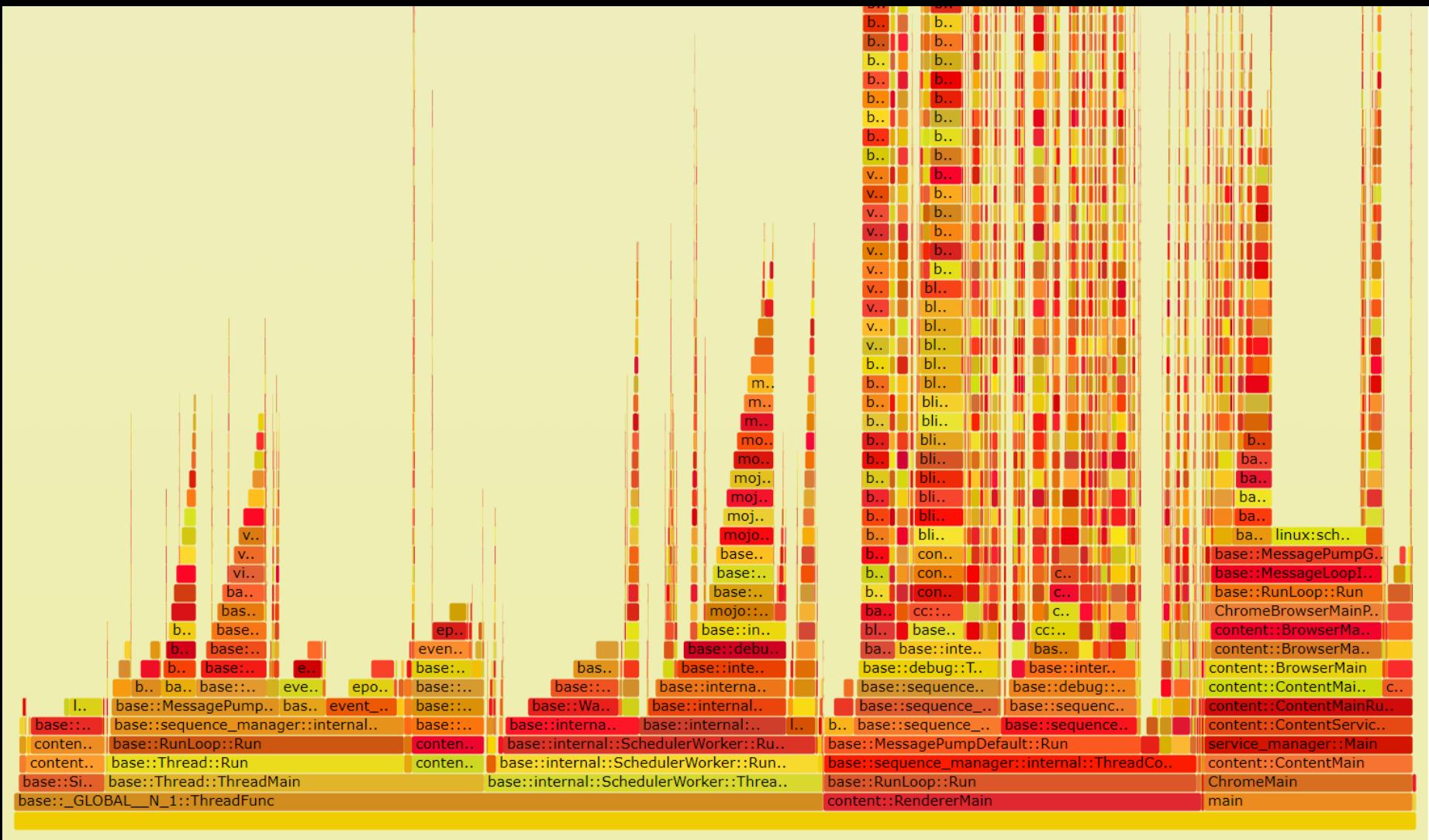
Flame Graph Output

```
$ uftrace dump --flame-graph | flamegraph.pl > abc.svg
```



<https://uftrace.github.io/dump/gcc.svg>

FlameGraph of Chromium



<https://utrace.github.io/dump/chrome.svg>

Tracing Pre-Built Binaries

(without -pg compilation)

GCC 의 컴파일 절차

Tracing Pre-Built Binaries

```
$ /usr/bin/gcc
```

Tracing Pre-Built Binaries

```
$ /usr/bin/gcc hello.c
```

Tracing Pre-Built Binaries

```
$ uftrace /usr/bin/gcc hello.c
```

Tracing Pre-Built Binaries

```
$ uftrace /usr/bin/gcc hello.c
```

```
uftrace: /home/honggyu/work/uftrace/cmds/record.c:1477:check_binary
```

```
ERROR: Can't find 'mcount' symbol in the '/usr/bin/gcc'.
```

```
It seems not to be compiled with -pg or -finstrument-functions flag.
```

```
You can rebuild your program with it or use -P option for dynamic  
tracing.
```

Tracing Pre-Built Binaries

```
$ uftrace /usr/bin/gcc hello.c
```

```
uftrace: /home/honggyu/work/uftrace/cmds/record.c:1477:check_binary
```

```
ERROR: Can't find 'mcount' symbol in the '/usr/bin/gcc'.
```

```
It seems not to be compiled with -pg or -finstrument-functions flag.
```

```
You can rebuild your program with it or use -P option for dynamic  
tracing.
```

```
$ uftrace --force /usr/bin/gcc hello.c
```

--force 옵션을 사용해
라이브러리 tracing 가능

Tracing Pre-Built Binaries

```
$ uftrace --force /usr/bin/gcc hello.c
```

```
# DURATION      TID      FUNCTION
 9.100 us [ 1709] | malloc();
 1.100 us [ 1709] | malloc();
 1.100 us [ 1709] | malloc();
 2.200 us [ 1709] | __cxa_atexit();
 1.100 us [ 1709] | malloc();
 1.000 us [ 1709] | malloc();
 1.000 us [ 1709] | malloc();
 1.000 us [ 1709] | __cxa_atexit();
 8.700 us [ 1709] | malloc();
 1.800 us [ 1709] | strlen();
 1.800 us [ 1709] | sbrk();
 9.400 us [ 1709] | malloc();
26.700 us [ 1709] | memcpy();
24.200 us [ 1709] | malloc();
 1.600 us [ 1709] | malloc();
 1.400 us [ 1709] | malloc();
 1.200 us [ 1709] | malloc();
 1.200 us [ 1709] | free();
 1.900 us [ 1709] | realloc();
 7.900 us [ 1709] | __fsetlocking();
 1.200 us [ 1709] | __fsetlocking();
 1.100 us [ 1709] | __fsetlocking();
                  [ 1709] | setlocale() {
 1.300 us [ 1709] |   free();
 ...

```

--force

Trace even if executable is
not instrumented

Tracing Pre-Built Binaries

```
$ uftrace --force -t 6ms /usr/bin/gcc hello.c
# DURATION      TID      FUNCTION
              [ 1702] | } = 0; /* vfork */
              [ 1702] | execv() {
488.109 ms  [ 1699] | waitpid();
              [ 1699] | vfork() {
              [ 1703] | } /* vfork */
              [ 1703] | execvp() {
8.926 ms   [ 1699] | } /* vfork */
147.425 ms  [ 1699] | waitpid();
              [ 1704] | } /* vfork */
              [ 1704] | execv() {
              [ 1699] | waitpid() {
              [ 1705] | } /* vfork */
              [ 1705] | execvp() {
151.569 ms  [ 1704] | waitpid();
253.676 ms  [ 1699] | } /* waitpid */
```

-t TIME, --time-filter=TIME

Do not show functions which run under the time threshold.

If some functions explicitly have the 'trace' trigger applied, those are always traced regardless of execution time.

Tracing Pre-Built Binaries

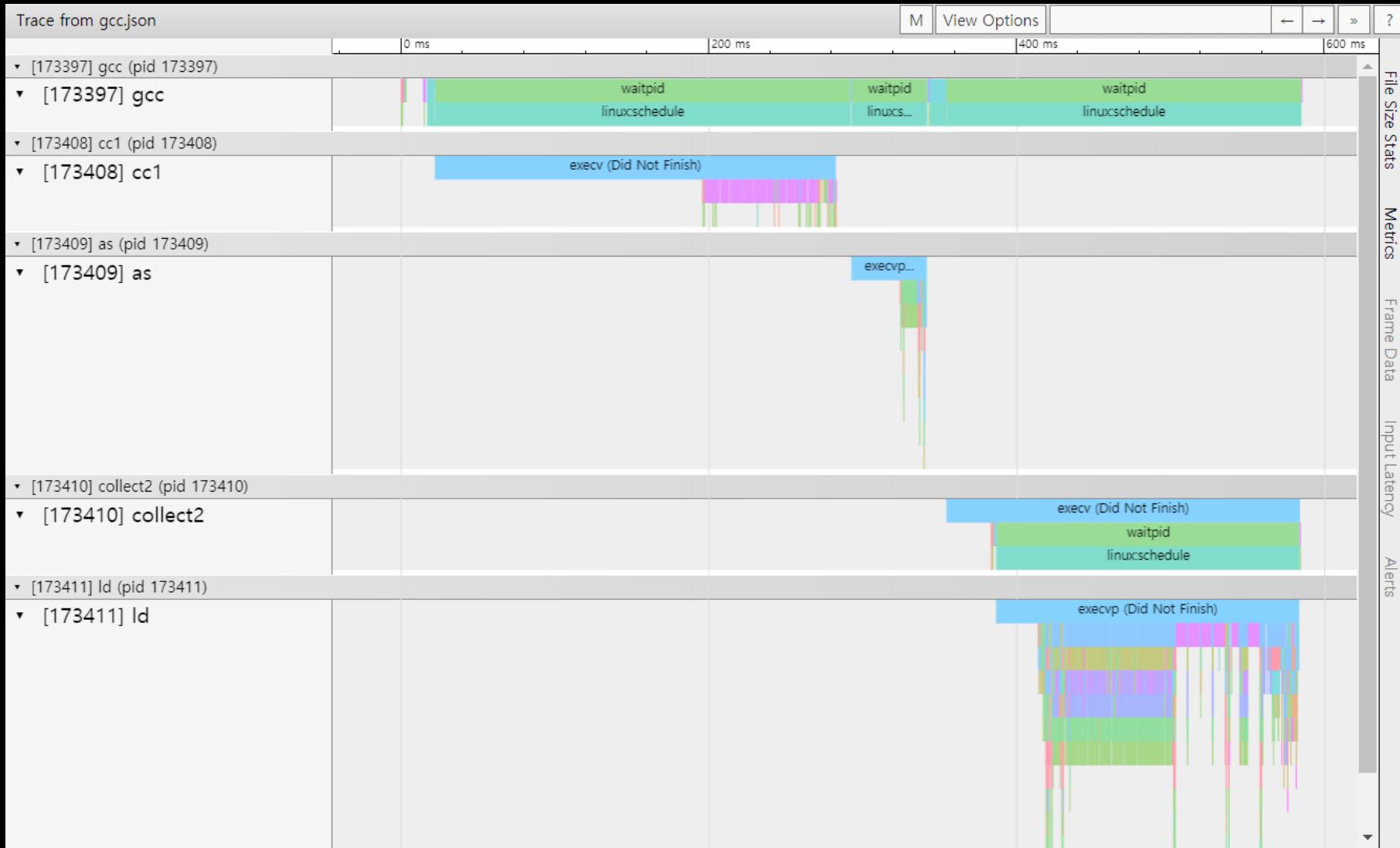
```
$ uftrace --force -t 6ms -a /usr/bin/gcc hello.c
# DURATION      TID      FUNCTION
              [ 1702] | } = 0; /* vfork */
              [ 1702] | execv("/usr/lib/gcc/x86_64-linux-gnu/7/cc1") {
488.109 ms  [ 1699] | waitpid(1702, 0x20aefa0, 0) = 1702;
              [ 1699] | vfork() {
              [ 1703] | } = 0; /* vfork */
              [ 1703] | execvp("as") {
8.926 ms   [ 1699] | } = 1703; /* vfork */
147.425 ms  [ 1699] | waitpid(1703, 0x20af090, 0) = 1703;
              [ 1704] | } = 0; /* vfork */
              [ 1704] | execv("/usr/lib/gcc/x86_64-linux-gnu/7/collect2") {
              [ 1699] | waitpid(1704, 0x20b65c0, 0) {
              [ 1705] | } = 0; /* vfork */
              [ 1705] | execvp("/usr/bin/ld") {
151.569 ms  [ 1704] | waitpid(1705, 0x16ab1b0, 0) = 1705;
253.676 ms  [ 1699] | } = 1704; /* waitpid */
```

-a, --auto-args

Automatically record arguments and return values of known functions. These are usually functions in standard (C language or system) libraries but if debug info is available it includes functions in the user program.

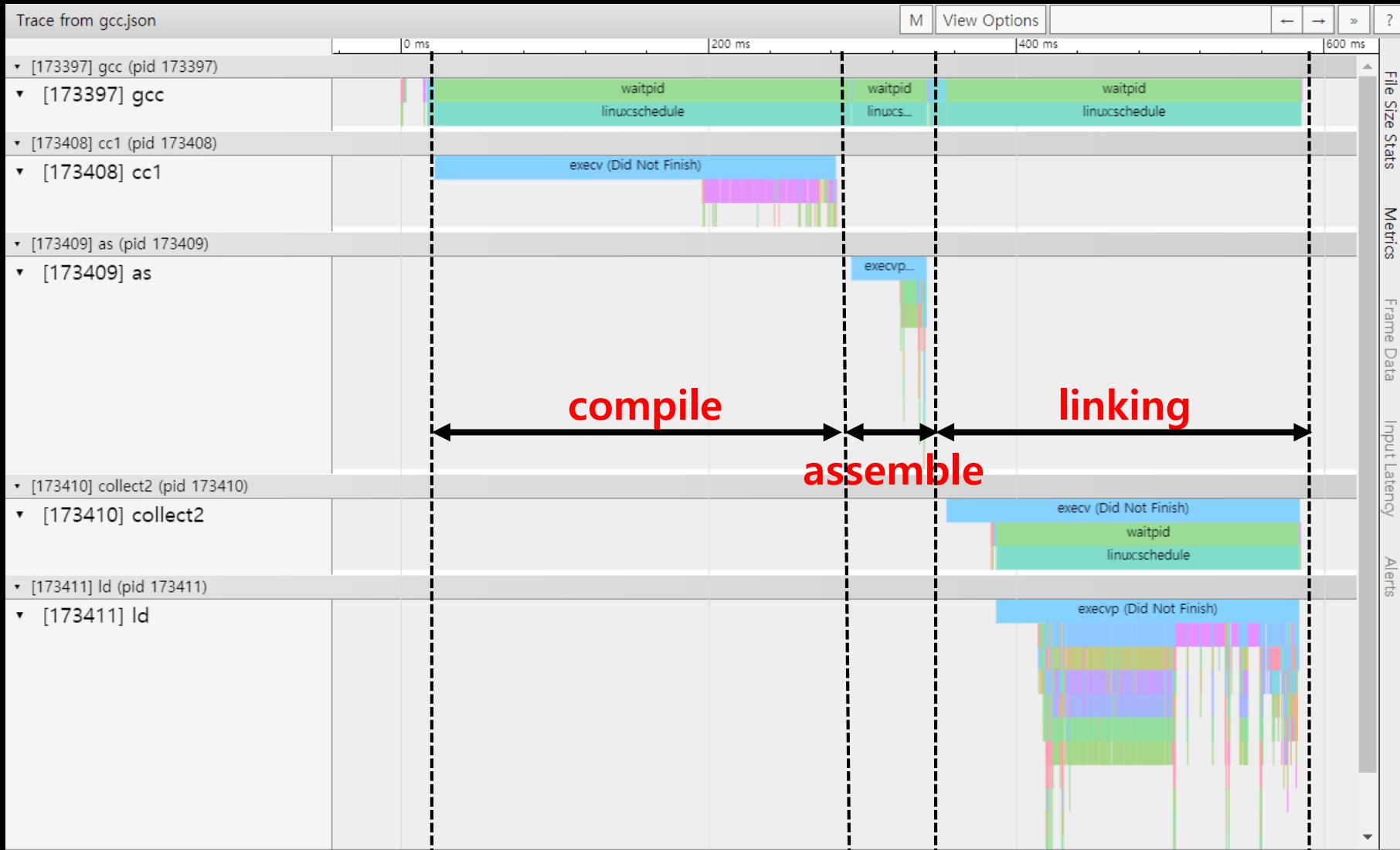
Tracing Pre-Built Binaries

```
$ uctrace dump --chrome
```



Tracing Pre-Built Binaries

```
$ uctrace dump --chrome
```



Task-Level Graph

- `uftrace` 는 기본적으로 function-level 실행 흐름을 기록
- 별도의 `--task` 옵션으로 task-level 실행 흐름 분석도 가능

```
$ uftrace record -la gcc hello.c
```

Task-Level Graph

- **uftrace** 는 기본적으로 function-level 실행 흐름을 기록
- 별도의 **--task** 옵션으로 task-level 실행 흐름 분석도 가능

```
$ uftrace record -la gcc hello.c
```

```
$ uftrace graph --task
=====
 TASK GRAPH =====
# TOTAL TIME      SELF TIME          TID      TASK NAME
 585.244 ms     19.317 ms [173397] : gcc
                                         : |
 260.530 ms    260.521 ms [173408] : +----cc1
                                         : |
 48.917 ms     48.473 ms [173409] : +----as
                                         : |
 229.666 ms    32.431 ms [173410] : +----collect2
                                         : |
 196.814 ms   193.011 ms [173411] : +----ld
```

Task-Level Graph

- `uftrace` 는 기본적으로 function-level 실행 흐름을 기록
- 별도의 `--task` 옵션으로 task-level 실행 흐름 분석도 가능

```
$ make
gcc -c main.c -o main.o
gcc -c module1.c -o module1.o
gcc -c module2.c -o module2.o
gcc main.o module1.o module2.o -o program
```

Task-Level Graph

- `uftrace` 는 기본적으로 function-level 실행 흐름을 기록
- 별도의 `--task` 옵션으로 task-level 실행 흐름 분석도 가능

```
$ uftrace record -la make  
gcc -c main.c -o main.o  
gcc -c module1.c -o module1.o  
gcc -c module2.c -o module2.o  
gcc main.o module1.o module2.o -o program
```

```
$ uftrace record -la make
gcc -c main.c -o main.o
gcc -c module1.c -o module1.o
gcc -c module2.c -o module2.o
gcc main.o module1.o module2.o -o program
```

```
$ uftrace record -la make
gcc -c main.c -o main.o
gcc -c module1.c -o module1.o
gcc -c module2.c -o module2.o
gcc main.o module1.o module2.o -o program

$ uftrace graph --task
===== TASK GRAPH =====
# TOTAL TIME      SELF TIME      TID : TASK NAME
  5.528  s      37.159 ms  [154864] : make
                                         : |
  1.833  s      33.513 ms  [154878] : +---- gcc
                                         : |
  1.745  s      1.745  s  [154879] : |      +---- cc1
                                         : |
  49.922 ms     49.922 ms  [154880] : |      +---- as
                                         : |
  1.660  s      32.818 ms  [154881] : +---- gcc
                                         : |
  1.573  s      1.573  s  [154885] : |      +---- cc1
                                         : |
  49.870 ms     49.870 ms  [154887] : |      +---- as
                                         : |
  1.764  s      33.560 ms  [154888] : +---- gcc
                                         : |
  1.700  s      1.700  s  [154889] : |      +---- cc1
                                         : |
  25.567 ms     25.567 ms  [154891] : |      +---- as
                                         : |
  227.120 ms    32.903 ms  [154892] : +---- gcc
                                         : |
  193.374 ms    13.606 ms  [154893] :           +---- collect2
                                         : |
  179.551 ms    179.551 ms [154894] :           +---- ld
```

Tracing Pre-Built Binaries

```
$ uftrace -la /bin/dd if=/dev/zero of=out bs=1k count=3
...
29.792 us [ 1364] | open("/dev/zero", O_RDONLY) = 4;
 8.125 us [ 1364] | dup2(4, 0);
 0.625 us [ 1364] | __errno_location();
5.364 us [ 1364] | close(4) = 0;
 5.573 us [ 1364] | lseek(0, 0, SEEK_CUR) = 0;
224.218 us [ 1364] | open("out", O_TRUNC|O_CREAT|O_WRONLY) = 4;
 2.760 us [ 1364] | dup2(4, 1);
 0.677 us [ 1364] | __errno_location();
 2.917 us [ 1364] | close(4) = 0;
 5.312 us [ 1364] | clock_gettime(CLOCK_MONOTONIC, 0xfffffe3fee7b8) = 0;
17.604 us [ 1364] | malloc(9219) = 0xaaaad08250e0;
22.865 us [ 1364] | read(0, 0xaaaad0826000, 1024) = 1024;
65.468 us [ 1364] | write(1, 0xaaaad0826000, 1024) = 1024;
 5.312 us [ 1364] | read(0, 0xaaaad0826000, 1024) = 1024;
19.687 us [ 1364] | write(1, 0xaaaad0826000, 1024) = 1024;
 4.532 us [ 1364] | read(0, 0xaaaad0826000, 1024) = 1024;
17.447 us [ 1364] | write(1, 0xaaaad0826000, 1024) = 1024;
 9.063 us [ 1364] | close(0) = 0;
228.957 us [ 1364] | close(1) = 0;
...
```

C++ Case Study with `uftrace`

`uftrace` 를 활용한 C++ 내부에 대한 분석

std::endl

std::endl 을 사용하지 말아야 하는 이유

```
$ cat endl.cc
#include <iostream>
#include <vector>
#include <cstdlib>
using namespace std;

int main(int argc, char* argv[])
{
    int iter = atoi(argv[1]);
    vector<int> v(iter, 0);

    for (int i = 0; i < v.size(); i++)
        cout << v[i] << endl;
}
```

```
$ cat no-endl.cc
#include <iostream>
#include <vector>
#include <cstdlib>
using namespace std;

int main(int argc, char* argv[])
{
    int iter = atoi(argv[1]);
    vector<int> v(iter, 0);

    for (int i = 0; i < v.size(); i++)
        cout << v[i] << '\n';
}
```

```
$ gcc -O2 -o endl endl.cc  
$ time ./endl 50000000 > /dev/null
```

real	0m32.594s
user	0m19.433s
sys	0m13.161s

```
$ gcc -O2 -o no-endl no-endl.cc  
$ time ./no-endl 50000000 > /dev/null
```

real	0m4.250s
user	0m4.138s
sys	0m0.112s

std::endl

- std::endl 은 내부에서 어떤 일을 하는가?

```
std::cout << std::endl;
```

std::endl

- std::endl 은 내부에서 어떤 일을 하는가?

```
std::cout << std::endl;
```



```
std::cout << '\n' << std::flush;
```

```
$ cat test-endl.cpp
#include <iostream>
#include <vector>

int main()
{
    std::vector<int> v = { 10, 20, 30 };
    for (auto a : v)
        std::cout << a << std::endl;
}
```

```
$ g++ -O2 -pg -o endl test-endl.cpp
```

```
$ uftrace endl
```

```
10
```

```
20
```

```
30
```

```
$ uftrace endl
```

std::endl 호출시 반복된 flush 발생

```
10  
20  
30
```

#	DURATION	TID	FUNCTION
		[122233]	_GLOBAL__sub_I_main() {
156.463	us	[122233]	std::ios_base::Init::Init();
0.426	us	[122233]	__cxa_atexit();
164.050	us	[122233]	} /* _GLOBAL__sub_I_main */
		[122233]	main() {
2.336	us	[122233]	operator new();
21.507	us	[122233]	std::basic_ostream::operator<<();
15.691	us	[122233]	std::basic_ostream::put();
2.920	us	[122233]	std::basic_ostream::flush();
0.907	us	[122233]	std::basic_ostream::operator<<();
2.394	us	[122233]	std::basic_ostream::put();
0.344	us	[122233]	std::basic_ostream::flush();
0.313	us	[122233]	std::basic_ostream::operator<<();
2.064	us	[122233]	std::basic_ostream::put();
0.234	us	[122233]	std::basic_ostream::flush();
3.050	us	[122233]	operator delete();
58.105	us	[122233]	} = 0; /* main */
3.837	us	[122233]	std::ios_base::~Init();

```
$ cat test-no-endl.cpp
#include <iostream>
#include <vector>

int main()
{
    std::vector<int> v = { 10, 20, 30 };
    for (auto a : v)
        std::cout << a << '\n';
}

$ g++ -O2 -pg -o no-endl test-no-endl.cpp
```

```
$ uftrace no-endl
```

```
10
```

```
20
```

```
30
```

```
$ uftrace no-endl
```

```
10  
20  
30
```

#	DURATION	TID	FUNCTION
		[122346]	_GLOBAL__sub_I_main() {
170.023	us	[122346]	std::ios_base::Init::Init();
0.380	us	[122346]	__cxa_atexit();
178.013	us	[122346]	} /* _GLOBAL__sub_I_main */
		[122346]	main() {
2.184	us	[122346]	operator new();
14.537	us	[122346]	std::basic_ostream::operator<<();
16.204	us	[122346]	std::__ostream_insert();
0.720	us	[122346]	std::basic_ostream::operator<<();
2.430	us	[122346]	std::__ostream_insert();
0.353	us	[122346]	std::basic_ostream::operator<<();
2.150	us	[122346]	std::__ostream_insert();
3.170	us	[122346]	operator delete();
46.781	us	[122346]	} = 0; /* main */
5.284	us	[122346]	std::ios_base::Init::~Init();

불필요한 flush 없음

new[] and delete[]

new[] 가 반환하는 주소는 어디인가?

new and delete

```
$ cat new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};

void addr_of_p(void* p) {}

int main()
{
    Point* p1 = new Point;
    addr_of_p(p1);
    delete p1;
}
```

new and delete

```
$ cat new-delete.cc          $ uftrace -la -f none a.out
class Point {
    char buf[10];
public:
    Point()  {}
    ~Point() {}
};

void addr_of_p(void* p) {}

int main()
{
    Point* p1 = new Point;
    addr_of_p(p1);
    delete p1;
}
```

new and delete

```
$ cat new-delete.cc
class Point {
    char buf[10];
public:
    Point() {}
    ~Point() {}
};

void addr_of_p(void* p) {}

int main()
{
    Point* p1 = new Point;
    addr_of_p(p1);
    delete p1;
}
```

```
$ uftrace -la -f none a.out
main() {
    operator new(10) {
        malloc(10) = 0x262af0;
    } = 0x262af0; /* operator new */
    Point::Point(0x262af0);
    addr_of_p(0x262af0);
    Point::~Point(0x262af0);
    operator delete(0x262af0) {
        free(0x262af0);
    } /* operator delete */
} = 0; /* main */
```

new[] and delete[]

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};

void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

new[] and delete[]

```
$ cat array-new-delete.cc      $ uftrace -la -f none a.out
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};

void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

new[] and delete[]

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() {}
    ~Point() {}
};

void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

```
$ uftrace -la -f none a.out
main() {
    operator new[] (38) {
        operator new(38) {
            malloc(38) = 0xd7a020;
        } = 0xd7a020; /* operator new */
    } = 0xd7a020; /* operator new[] */
    Point::Point(0xd7a028);
    Point::Point(0xd7a032);
    Point::Point(0xd7a03c);
    addr_of_p(0xd7a028);
    Point::~Point(0xd7a03c);
    Point::~Point(0xd7a032);
    Point::~Point(0xd7a028);
    operator delete[] (0xd7a020) {
        operator delete(0xd7a020) {
            free(0xd7a020);
        } /* operator delete */
    } /* operator delete[] */
} = 0; /* main */
```

new[] and delete[]

10 바이트 크기의 객체 3개
를 할당했는데 왜 38 바이트 할당을 요청할까?

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() {}
    ~Point() {}
};

void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

```
$ uftrace -la -f none a.out
main() {
    operator new[] (38) ←{
        operator new(38) {
            malloc(38) = 0xd7a020;
        } = 0xd7a020; /* operator new */
        } = 0xd7a020; /* operator new[] */
    Point::Point(0xd7a028);
    Point::Point(0xd7a032);
    Point::Point(0xd7a03c);
    addr_of_p(0xd7a028);
    Point::~Point(0xd7a03c);
    Point::~Point(0xd7a032);
    Point::~Point(0xd7a028);
    operator delete[] (0xd7a020) {
        operator delete(0xd7a020) {
            free(0xd7a020);
        } /* operator delete */
    } /* operator delete[] */
} = 0; /* main */
```

new[] and delete[]

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() {}
    ~Point() {}
};

void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

왜 실제 메모리가 할당된
주소보다 80이 클까?

```
$ uftrace -la -f none a.out
main() {
    operator new[] (38) ←{
        operator new(38) {
            malloc(38) = 0xd7a020;
        } = 0xd7a020; /* operator new */
        } = 0xd7a020; /* operator new[] */
    Point::Point(0xd7a028);
    Point::Point(0xd7a032);
    Point::Point(0xd7a03c);
    addr_of_p(0xd7a028);
    Point::~Point(0xd7a03c);
    Point::~Point(0xd7a032);
    Point::~Point(0xd7a028);
    operator delete[](0xd7a020) {
        operator delete(0xd7a020) {
            free(0xd7a020);
        } /* operator delete */
    } /* operator delete[] */
} = 0; /* main */
```

10 바이트 크기의 객체 3개
를 할당했는데 왜 38 바이트
할당을 요청할까?

new[] and delete[]

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};
```

```
void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

```
$ cat wrong-array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};
```

```
void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

new[] and **deleteX**

```
$ cat array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};
```

```
void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    delete[] p2;
}
```

```
$ cat wrong-array-new-delete.cc
class Point {
    char buf[10];
public:
    Point() { }
    ~Point() { }
};
```

```
void addr_of_p(void* p) {}

int main()
{
    Point* p2 = new Point[3];
    addr_of_p(p2);
    // delete[] p2;
    delete p2;
}
```

new[] and delete

```
$ uftrace -la wrong-array-new-delete
```

new[] and delete

```
$ uftrace -la wrong-array-new-delete
*** Error in `wrong-array-new-delete': munmap_chunk(): invalid pointer: 0x1af0aa8 ***
process crashed by signal 11: Segmentation fault (si_code: 128)
child terminated by signal: 11: Segmentation fault
```

new[] and delete

```
$ uftrace -la wrong-array-new-delete
*** Error in `wrong-array-new-delete': munmap_chunk(): invalid pointer: 0x1af0aa8 ***
process crashed by signal 11: Segmentation fault (si_code: 128)
child terminated by signal: 11: Segmentation fault
# DURATION      TID      FUNCTION
              [ 79033] | main() {
              [ 79033] |   operator new[] (38) {
              [ 79033] |     operator new(38) {
0.933 us  [ 79033] |       malloc(38) = 0x1af0aa0;
4.687 us  [ 79033] |     } = 0x1af0aa0; /* operator new */
12.374 us  [ 79033] |   } = 0x1af0aa0; /* operator new[] */
0.460 us  [ 79033] |     Point::Point(0x1af0aa8);
0.210 us  [ 79033] |     Point::Point(0x1af0ab2);
0.207 us  [ 79033] |     Point::Point(0x1af0abc);
0.227 us  [ 79033] |     addr_of_p(0x1af0aa8);
0.230 us  [ 79033] |     Point::~Point(0x1af0aa8);
              [ 79033] |   operator delete(0x1af0aa8) {
              [ 79033] |     free(0x1af0aa8) {
              [ 79033] |   /* linux:task-exit */

uftrace stopped tracing with remaining functions
=====
task: 79033
[2] free
[1] operator delete
[0] main
```

new[] and delete

```
$ uftrace -la wrong-array-new-delete
*** Error in `wrong-array-new-delete': munmap_chunk(): invalid pointer: 0x1af0aa8 ***
process crashed by signal 11: Segmentation fault (si_code: 128)
child terminated by signal: 11: Segmentation fault
# DURATION      TID      FUNCTION
[ 79033] | main() {
[ 79033] |     operator new[] (38) {
[ 79033] |         operator new(38) {
0.933 us [ 79033] |             malloc(38) = 0x1af0aa0;
4.687 us [ 79033] |         } = 0x1af0aa0; /* operator new */
12.374 us [ 79033] |     } = 0x1af0aa0; /* operator new[] */
0.460 us [ 79033] |     Point::Point(0x1af0aa8);
0.210 us [ 79033] |     Point::Point(0x1af0ab2);
0.207 us [ 79033] |     Point::Point(0x1af0abc);
0.227 us [ 79033] |     addr_of_p(0x1af0aa8);
0.230 us [ 79033] |     Point::~Point(0x1af0aa8);
[ 79033] |     operator delete(0x1af0aa8) {
[ 79033] |         free(0x1af0aa8) {
[ 79033] |             /* linux:task-exit */

uftrace stopped tracing with remaining functions
=====
task: 79033
[2] free
[1] operator delete
[0] main
```

실제 malloc 으로 얻은 주소와 free 를 요청하는 주소가 달라서 segfault

new[] and delete

```
$ uftrace -la wrong-array-new-delete
*** Error in `wrong-array-new-delete': munmap_chunk(): invalid pointer: 0x1af0aa8 ***
process crashed by signal 11: Segmentation fault (si_code: 128)
child terminated by signal: 11: Segmentation fault
# DURATION      TID      FUNCTION
[ 79033] | main() {
[ 79033] |     operator new[] (38) {
[ 79033] |         operator new(38) {
0.933 us [ 79033] |             malloc(38) = 0x1af0aa0;
4.687 us [ 79033] |         } = 0x1af0aa0; /* operator new */
12.374 us [ 79033] |     } = 0x1af0aa0; /* operator new[] */
0.460 us [ 79033] |     Point::Point(0x1af0aa8);
0.210 us [ 79033] |     Point::Point(0x1af0ab2);
0.207 us [ 79033] |     Point::Point(0x1af0abc);
0.227 us [ 79033] |     addr_of_p(0x1af0aa8);
0.230 us [ 79033] |     Point::~Point(0x1af0aa8);
[ 79033] |     operator delete(0x1af0aa8) {
[ 79033] |         free(0x1af0aa8) {
[ 79033] |             /* linux:task-exit */

실제 malloc 으로 얻은
주소와 free 를 요청하는
주소가 달라서 segfault

uftrace stopped tracing with remaining functions
=====
task: 79033
[2] free
[1] operator delete
[0] main
segfault 로 프로그램이 비정상
종료할 때까지의 경로도 확인 가능!
```

new[] and delete[]

```
Point* p = new Point[3]
```

new[] and delete[]

```
Point* p = new Point[3]
```

Point[0]

Point[1]

Point[2]

new[] and delete[]

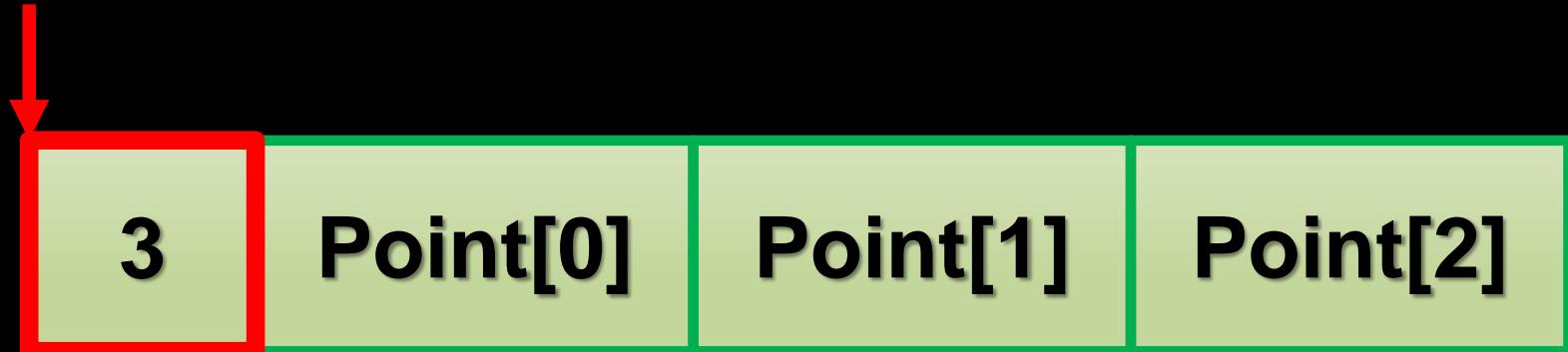
```
Point* p = new Point[3]
```



new[] and delete[]

```
Point* p = new Point[3]
```

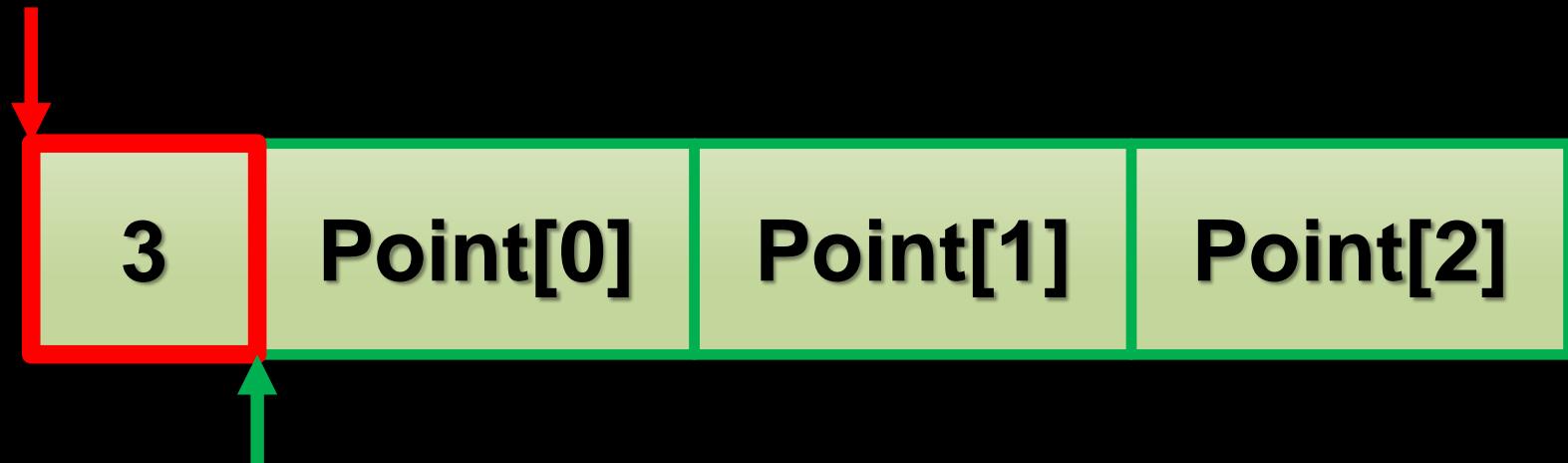
실제 내부적으로 할당하는 메모리 공간의 시작 주소



new[] and delete[]

```
Point* p = new Point[3]
```

실제 내부적으로 할당하는 메모리 공간의 시작 주소

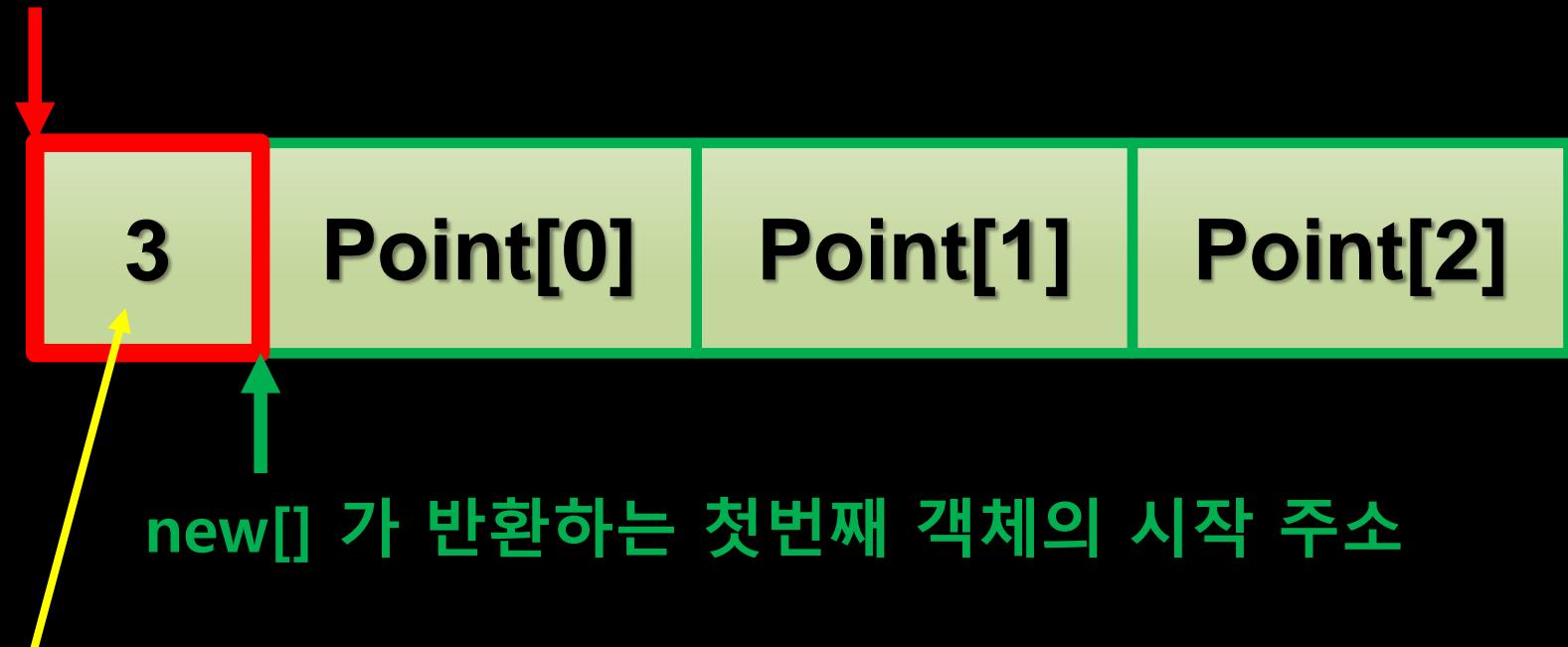


new[] 가 반환하는 첫번째 객체의 시작 주소

new[] and delete[]

```
Point* p = new Point[3]
```

실제 내부적으로 할당하는 메모리 공간의 시작 주소



new[] 가 반환하는 첫번째 객체의 시작 주소

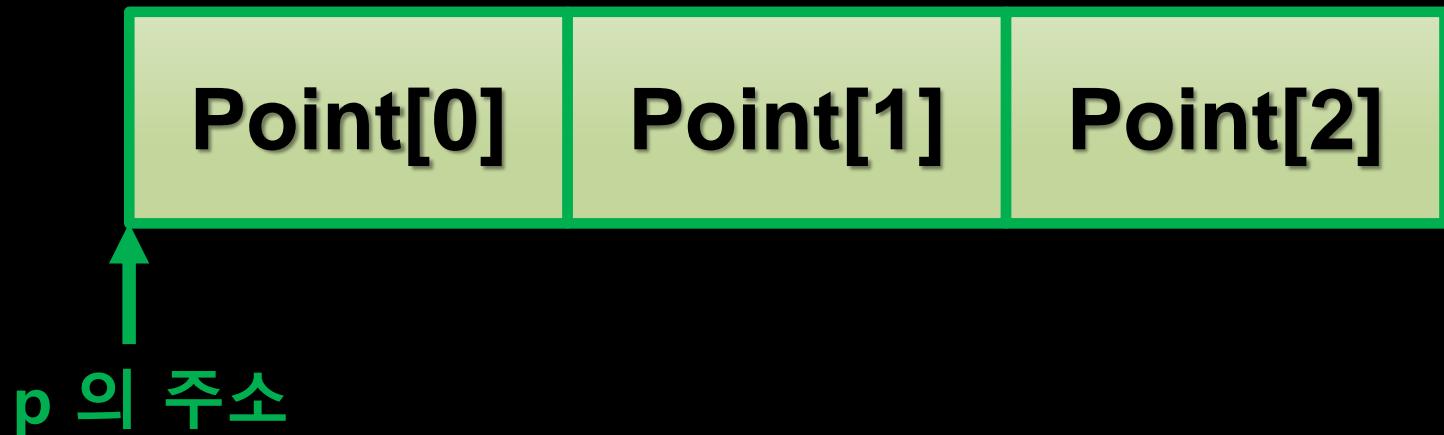
할당된 객체의 수

new[] and delete[]

delete [] p

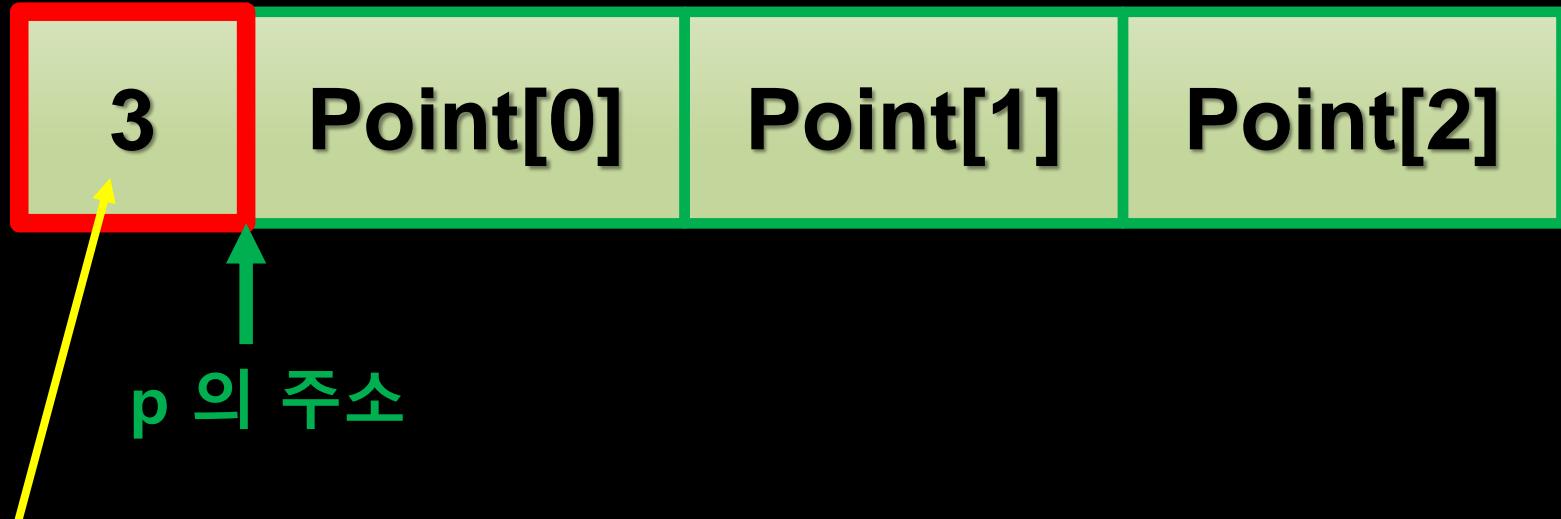
new[] and delete[]

delete [] p



new[] and delete[]

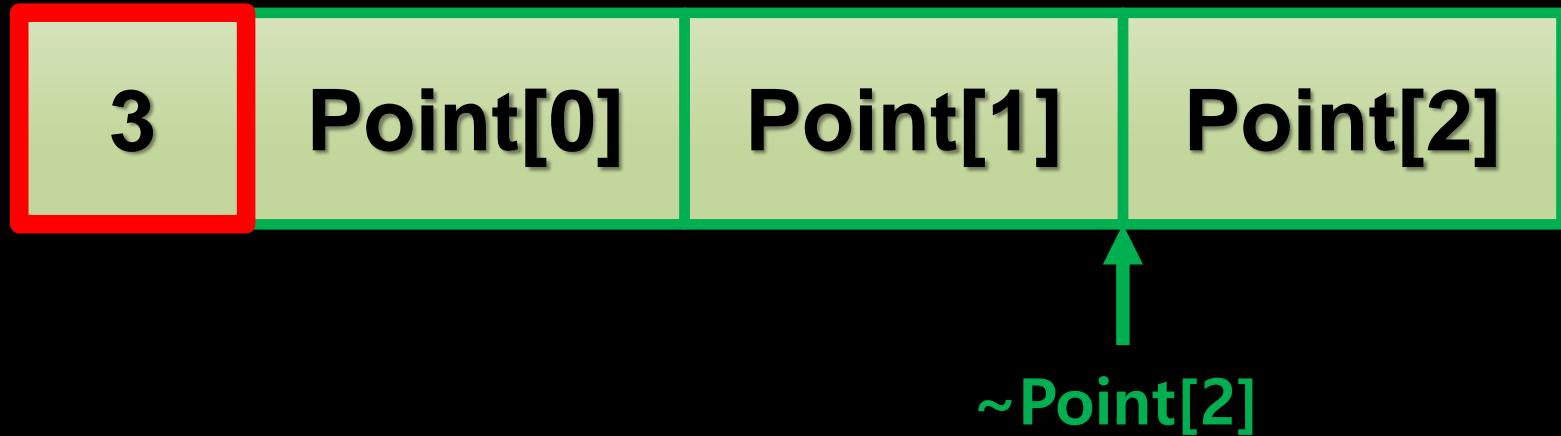
delete [] p



할당된 객체의 수

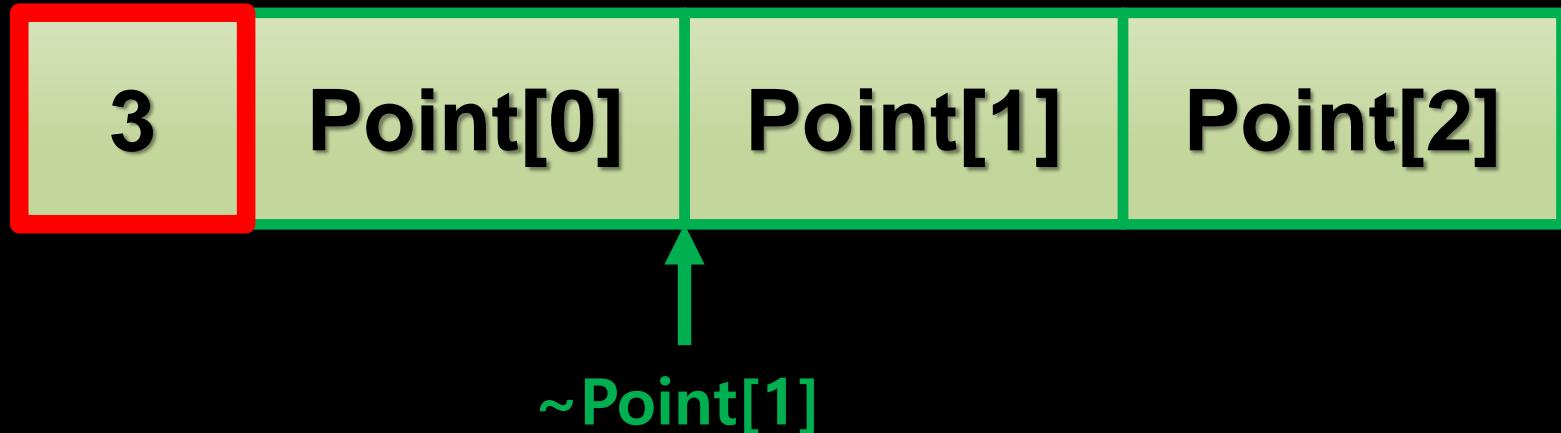
new[] and delete[]

delete [] p



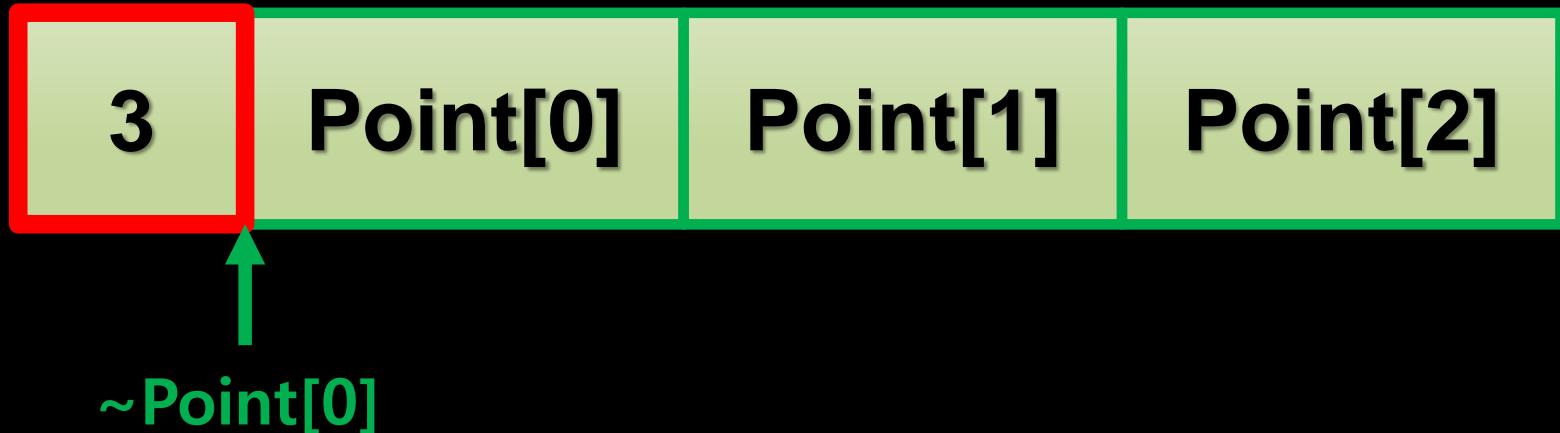
new[] and delete[]

delete [] p



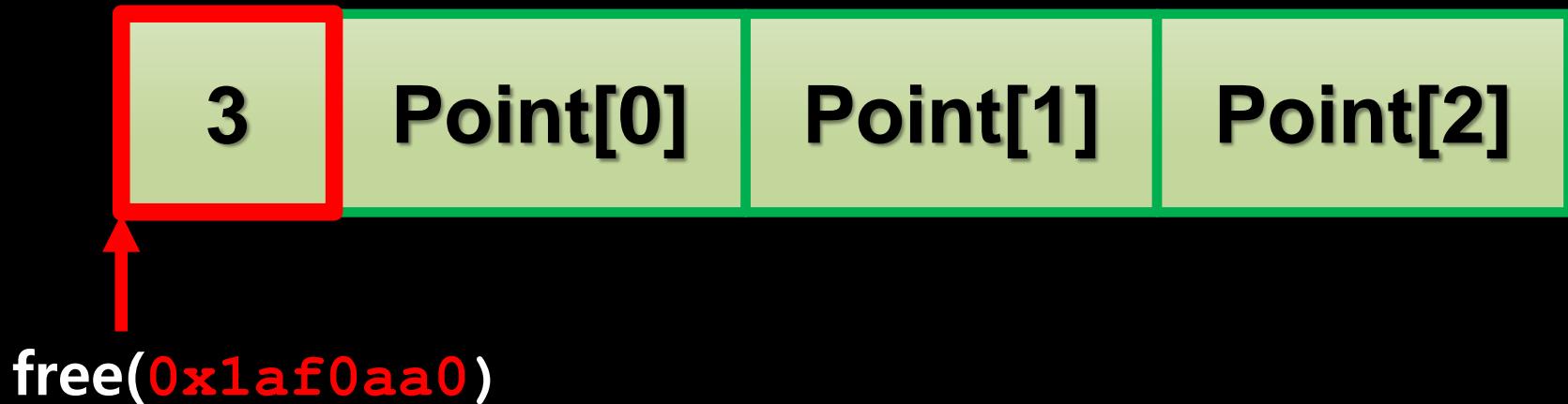
new[] and delete[]

delete [] p



new[] and delete[]

delete [] p



new[] and delete[]

delete [] p



free(0x1af0aa0)

new[] and delete[]

delete [] p

정상적인 메모리 해제



free(0x1af0aa0)

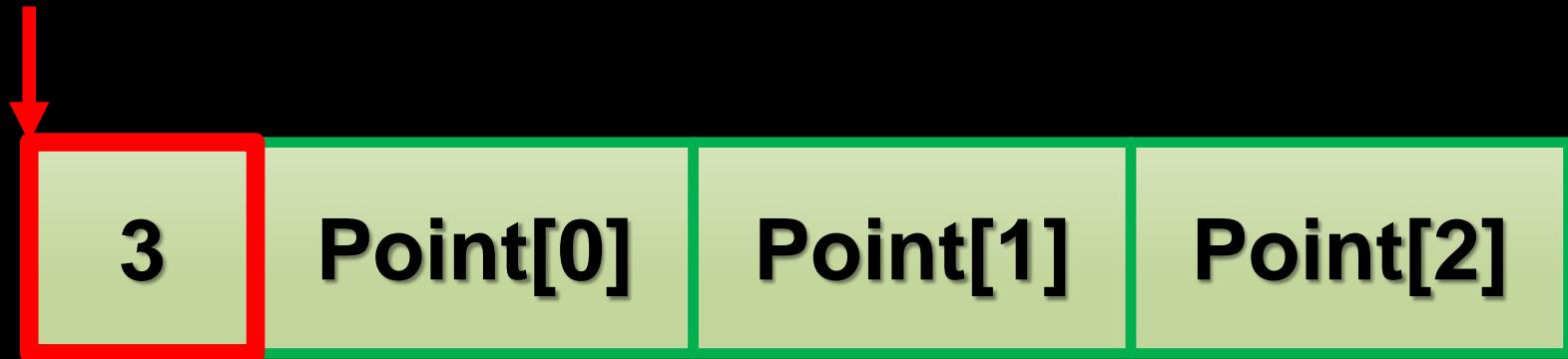
new[] and delete[]

```
main() {  
    operator new[] (38) {  
        operator new(38) {  
            malloc(38) = 0xd7a020;  
        } = 0xd7a020; /* operator new */  
    } = 0xd7a020; /* operator new[] */  
    Point::Point(0xd7a028);           Point[0]  
    Point::Point(0xd7a032);           Point[1]  
    Point::Point(0xd7a03c);           Point[2]  
    addr_of_p(0xd7a028);  
    Point::~Point(0xd7a03c);         ~Point[2]  
    Point::~Point(0xd7a032);         ~Point[1]  
    Point::~Point(0xd7a028);         ~Point[0]  
    operator delete[] (0xd7a020) {  
        operator delete(0xd7a020) {  
            free(0xd7a020);  
        } /* operator delete */  
    } /* operator delete[] */  
} = 0; /* main */
```

new[] and delete[]

```
Point* p = new Point[3]
```

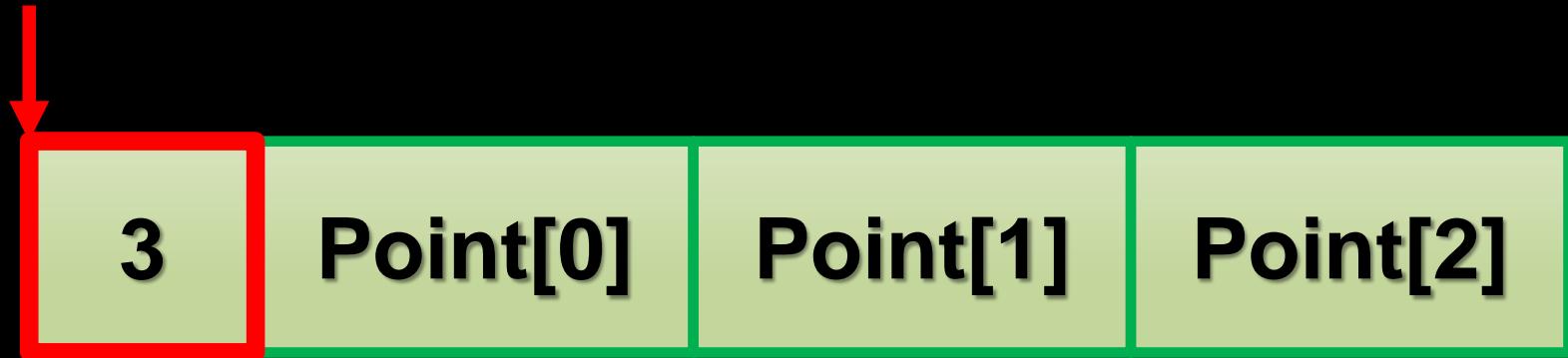
```
malloc(38) = 0x1af0aa0;
```



new[] and **deleteX**

```
Point* p = new Point[3]  
delete p;
```

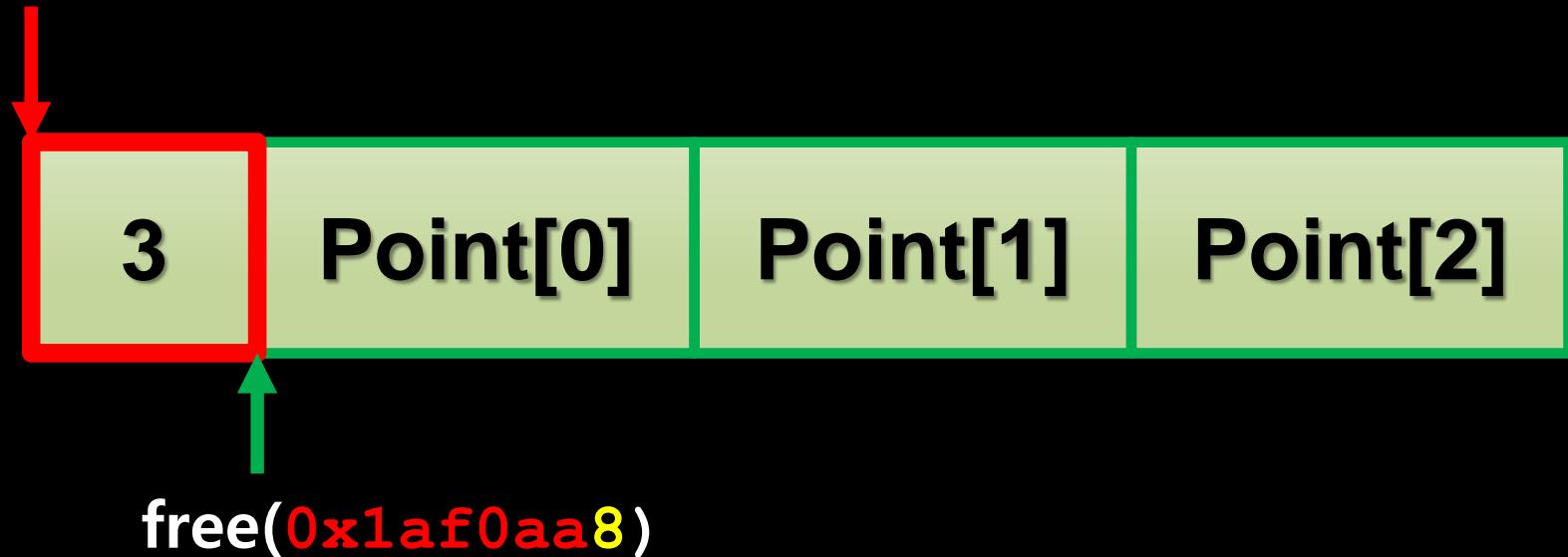
malloc(38) = 0x1af0aa0;



`new[]` and `deleteX`

```
Point* p = new Point[3]  
delete p;
```

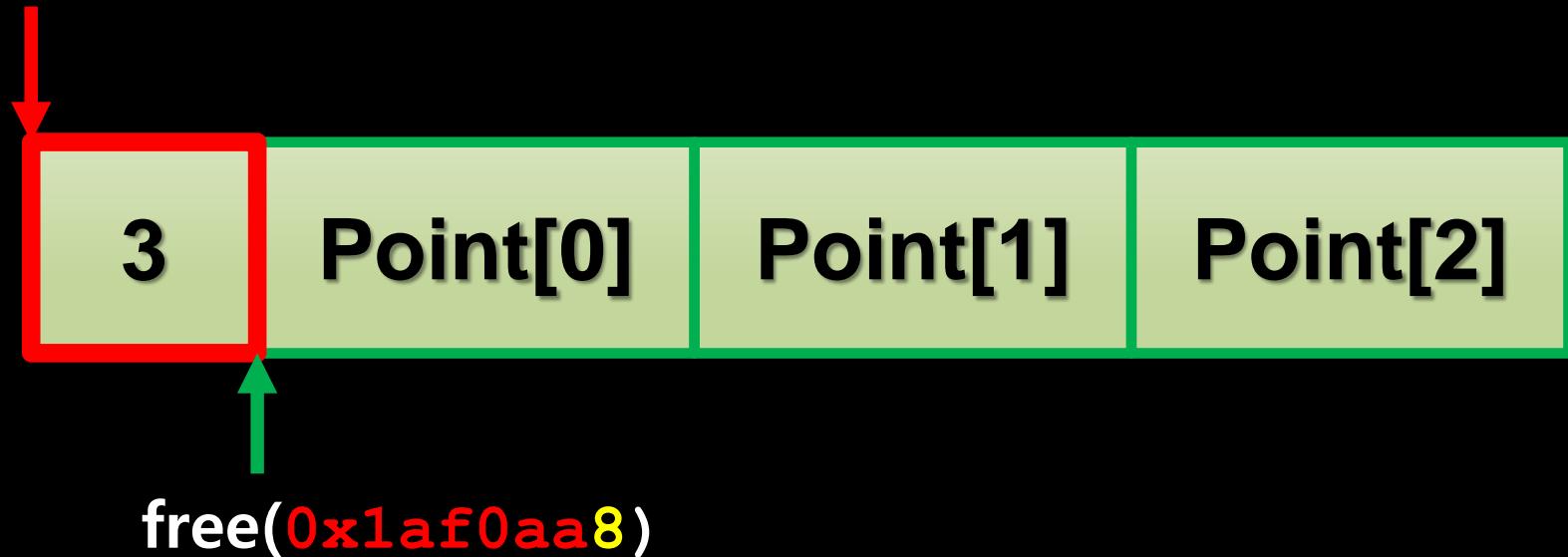
`malloc(38) = 0x1af0aa0;`



`new[]` and `delete X`

```
Point* p = new Point[3]  
delete p;
```

`malloc(38) = 0x1af0aa0;`



할당되지 않은 주소에 대한 메모리 해제 요청

Zero-Cost Abstractions

Abstractions have cost, but mostly zero in C++!

std::unique_ptr

new / delete 와 성능 차이가 있을까?

new and delete

```
$ cat new-delete.cc
int main()
{
    int* p = new int;
    delete p;
}
```

new and delete

```
$ cat new-delete.cc
int main()
{
    int* p = new int;
    delete p;
}

$ g++ -pg -g -O2 new-delete.cc
```

new and delete

```
$ cat new-delete.cc
int main()
{
    int* p = new int;
    delete p;
}
```

```
$ g++ -pg -g -O2 new-delete.cc
```

```
$ uftrace -la a.out
# DURATION      TID      FUNCTION
                           [165267]  | main() {
                           [165267]  |     operator new(4)  {
0.976 us  [165267]  |         malloc(4)  = 0x21ceaa0;
7.847 us  [165267]  |     } = 0x21ceaa0; /* operator new */
                           [165267]  |     operator delete(0x21ceaa0) {
1.807 us  [165267]  |         free(0x21ceaa0);
5.261 us  [165267]  |     } /* operator delete */
18.154 us  [165267]  | } = 0; /* main */
```

std::unique_ptr

```
$ cat unique_ptr.cc
#include <memory>
int main()
{
    std::unique_ptr<int> p(new int);
}
```

std::unique_ptr

```
$ cat unique_ptr.cc
#include <memory>
int main()
{
    std::unique_ptr<int> p(new int);
}
```



```
$ g++ -std=c++11 -pg -g -O2 unique_ptr.cc
```

std::unique_ptr

```
$ cat unique_ptr.cc
#include <memory>
int main()
{
    std::unique_ptr<int> p(new int);
}

$ g++ -std=c++11 -pg -g -O2 unique_ptr.cc

$ uftrace -la a.out
# DURATION      TID      FUNCTION
                           [165318]  | main() {
                           [165318]  |     operator new(4)  {
1.076 us  [165318]  |         malloc(4) = 0x1d7b290;
7.824 us  [165318]  |     } = 0x1d7b290; /* operator new */
                           [165318]  |     operator delete(0x1d7b290) {
                           [165318]  |         free(0x1d7b290);
                           [165318]  |     } /* operator delete */
18.260 us  [165318]  | } = 0; /* main */
```

std::unique_ptr

```
$ cat unique_ptr.cc
#include <memory>
int main()
{
    std::unique_ptr<int> p(new int);
}
```

```
$ g++ -std=c++11 -pg -g -O2 unique_ptr.cc
```

```
$ uftrace -la a.out
```

#	DURATION	TID	FUNCTION
		[165318]	main() {
		[165318]	operator new(4) {
1.076 us		[165318]	malloc(4) = 0x1d7b290;
7.824 us		[165318]	} = 0x1d7b290; /* operator new */
		[165318]	operator delete(0x1d7b290) {
1.716 us		[165318]	free(0x1d7b290);
5.097 us		[165318]	} /* operator delete */
18.260 us		[165318]	} = 0; /* main */

new 와 delete 를 직접
사용한 경우와 동일함

#	DURATION	TID	FUNCTION
		[168276]	main() {
		[168276]	operator new(4) {
		[168276]	malloc(4) = 0xbdbb40;
1.094 us		[168276]	} = 0xbdbb40; /* operator new */
8.114 us		[168276]	std::unique_ptr::unique_ptr(0x7ffdःd6fd110, 0xbdbb40) {
		[168276]	std::tuple::tuple(0x7ffdःd6fd110) {
		[168276]	std::tuple_implementation::tuple_implementation(0x7ffdःd6fd110) {
		[168276]	std::head_base::head_base(0x7ffdःd6fd110) {
0.266 us		[168276]	std::default_delete::default_delete(0x7ffdःd6fd110);
1.674 us		[168276]	} /* std::head_base::head_base */
6.320 us		[168276]	} /* std::tuple_implementation::tuple_implementation */
0.227 us		[168276]	std::head_base::head_base(0x7ffdःd6fd110);
7.850 us		[168276]	} /* std::head_base::head_base */
8.500 us		[168276]	} /* std::tuple::tuple */
		[168276]	std::get(0x7ffdःd6fd110) {
		[168276]	std::__get_helper(0x7ffdःd6fd110) {
		[168276]	std::tuple_implementation::M_head(0x7ffdःd6fd110) {
0.226 us		[168276]	std::head_base::M_head(0x7ffdःd6fd110) = 0x7ffdःd6fd110;
1.290 us		[168276]	} = 0x7ffdःd6fd110; /* std::tuple_implementation::M_head */
2.113 us		[168276]	} = 0x7ffdःd6fd110; /* std::__get_helper */
2.816 us		[168276]	} = 0x7ffdःd6fd110; /* std::get */
12.870 us		[168276]	} /* std::unique_ptr::unique_ptr */
		[168276]	std::unique_ptr::~unique_ptr(0x7ffdःd6fd110) {
		[168276]	std::get(0x7ffdःd6fd110) {
		[168276]	std::__get_helper(0x7ffdःd6fd110) {
0.210 us		[168276]	std::tuple_implementation::M_head(0x7ffdःd6fd110) {
1.093 us		[168276]	std::head_base::M_head(0x7ffdःd6fd110) = 0x7ffdःd6fd110;
1.646 us		[168276]	} = 0x7ffdःd6fd110; /* std::tuple_implementation::M_head */
2.220 us		[168276]	} = 0x7ffdःd6fd110; /* std::__get_helper */
		[168276]	} = 0x7ffdःd6fd110; /* std::get */
		[168276]	std::unique_ptr::get_deleter(0x7ffdःd6fd110) {
		[168276]	std::get(0x7ffdःd6fd110) {
		[168276]	std::__get_helper(0x7ffdःd6fd110) {
0.213 us		[168276]	std::tuple_implementation::M_head(0x7ffdःd6fd110) {
1.094 us		[168276]	std::head_base::M_head(0x7ffdःd6fd110) = 0x7ffdःd6fd110;
1.747 us		[168276]	} = 0x7ffdःd6fd110; /* std::tuple_implementation::M_head */
2.391 us		[168276]	} = 0x7ffdःd6fd110; /* std::__get_helper */
3.223 us		[168276]	} = 0x7ffdःd6fd110; /* std::get */
		[168276]	} = 0x7ffdःd6fd110; /* std::unique_ptr::get_deleter */
		[168276]	std::default_delete::operator()(0x7ffdःd6fd110, 0xbdbb40) {
		[168276]	operator delete(0xbdbb40) {
1.810 us		[168276]	free(0xbdbb40);
5.230 us		[168276]	} /* operator delete */
6.213 us		[168276]	} /* std::default_delete::operator() */
13.271 us		[168276]	} /* std::unique_ptr::~unique_ptr */
40.624 us		[168276]	} = 0; /* main */

최적화를 안한 경우의 전체 내부 함수 호출

```

# DURATION      TID      FUNCTION
[168276] | main() {
[168276] |     operator new(4) {
[168276] |         malloc(4) = 0xbdbb40;
1.094 us [168276] |     } = 0xbdbb40; /* operator new */
8.114 us [168276] |     std::unique_ptr::unique_ptr(0x7ffd6fd110, 0xbdbb40) {
[168276] |         std::tuple::tuple(0x7ffd6fd110) {
[168276] |             std::_Tuple_impl::_Tuple_impl(0x7ffd6fd110) {
[168276] |                 std::_Tuple_impl::_Tuple_impl(0x7ffd6fd110) {
[168276] |                     std::_Head_base::_Head_base(0x7ffd6fd110) {
[168276] |                         std::default_delete::default_delete(0x7ffd6fd110);
0.266 us [168276] |                     } /* std::_Head_base::_Head_base */
1.674 us [168276] |                 } /* std::_Tuple_impl::_Tuple_impl */
6.320 us [168276] |                 std::_Head_base::_Head_base(0x7ffd6fd110);
0.227 us [168276] |             } /* std::_Tuple_impl::_Tuple_impl */
7.850 us [168276] |         } /* std::tuple */
8.500 us [168276] |     } /* std::unique_ptr */
[168276] |     std::get(0x7ffd6fd110) {
[168276] |         std::__get_helper(0x7ffd6fd110) {
[168276] |             std::_Tuple_impl::_M_head(0x7ffd6fd110) {
0.226 us [168276] |                 std::_Head_base::_M_head(0x7ffd6fd110) = 0x7ffd6fd110;
1.290 us [168276] |             } = 0x7ffd6fd110; /* std::_Tuple_impl::_M_head */
2.113 us [168276] |             } = 0x7ffd6fd110; /* std::__get_helper */
2.816 us [168276] |             } = 0x7ffd6fd110; /* std::get */
12.870 us [168276] |         } /* std::unique_ptr */
[168276] |         std::unique_ptr::~unique_ptr(0x7ffd6fd110) {
[168276] |             std::get(0x7ffd6fd110) {
[168276] |                 std::__get_helper(0x7ffd6fd110) {
[168276] |                     std::_Tuple_impl::_M_head(0x7ffd6fd110) {
0.210 us [168276] |                         std::_Head_base::_M_head(0x7ffd6fd110) = 0x7ffd6fd110;
1.093 us [168276] |                     } = 0x7ffd6fd110; /* std::_Tuple_impl::_M_head */
1.646 us [168276] |                     } = 0x7ffd6fd110; /* std::__get_helper */
2.220 us [168276] |                     } = 0x7ffd6fd110; /* std::get */
[168276] |                     std::unique_ptr::get_deleter(0x7ffd6fd110) {
[168276] |                         std::get(0x7ffd6fd110) {
[168276] |                             std::__get_helper(0x7ffd6fd110) {
[168276] |                                 std::_Tuple_impl::_M_head(0x7ffd6fd110) {
0.213 us [168276] |                                     std::_Head_base::_M_head(0x7ffd6fd110) = 0x7ffd6fd110;
1.094 us [168276] |                                     } = 0x7ffd6fd110; /* std::_Tuple_impl::_M_head */
1.747 us [168276] |                                     } = 0x7ffd6fd110; /* std::__get_helper */
2.391 us [168276] |                                     } = 0x7ffd6fd110; /* std::get */
3.223 us [168276] |                                     } = 0x7ffd6fd110; /* std::unique_ptr::get_deleter */
[168276] |                         std::default_delete::operator()(0x7ffd6fd110, 0xbdbb40) {
[168276] |                             operator delete(0xbdbb40) {
1.810 us [168276] |                                 free(0xbdbb40);
5.230 us [168276] |                             } /* operator delete */
6.213 us [168276] |                         } /* std::default_delete::operator() */
13.271 us [168276] |                     } /* std::unique_ptr::~unique_ptr */
40.624 us [168276] |                 } = 0; /* main */

```

qsort vs. std::sort

C 언어 libc 의 qsort 와
C++ algorithm 의 std::sort 비교

qsort

```
#include <stdlib.h>

int qcomp(const void *a, const void *b)
{
    return *(int*)a - *(int*)b;
}

__attribute__((noinline))
void test_qsort()
{
    int arr[] = { 4, 1, 3, 7, 2, 6, 5 };
    qsort(arr, 7, sizeof(int), qcomp);
}

int main()
{
    test_qsort();
}
```

qsort

```
$ uftrace qsort
# DURATION      TID      FUNCTION
  1.189 us  [196210] | __monstartup();
  0.612 us  [196210] | __cxa_atexit();
                      [196210] | main() {
                      [196210] |     test_qsort() {
                      [196210] |         qsort() {
  0.138 us  [196210] |             qcomp();
  0.101 us  [196210] |             qcomp();
  0.080 us  [196210] |             qcomp();
  0.097 us  [196210] |             qcomp();
  0.080 us  [196210] |             qcomp();
  0.097 us  [196210] |             qcomp();
  0.072 us  [196210] |             qcomp();
  0.096 us  [196210] |             qcomp();
  3.823 us  [196210] |         } /* qsort */
  4.322 us  [196210] |     } /* test_qsort */
  4.695 us  [196210] | } /* main */
```

std::sort (with function pointer)

```
#include <algorithm>
using namespace std;

int comp(int a, int b)
{
    return a < b;
}

__attribute__((noinline))
void test_std_sort()
{
    int arr[] = { 4, 1, 3, 7, 2, 6, 5 };
    std::sort(begin(arr), end(arr), comp);
}

int main()
{
    test_std_sort();
}
```

```
$ uftrace -T comp@arg1,arg2,retval std_sort
# DURATION      TID      FUNCTION
  1.850 us [196398] | __monstartup();
  0.690 us [196398] | __cxa_atexit();
[196398] | main() {
[196398] |     test_std_sort() {
  0.405 us [196398] |         std::__introsort_loop();
[196398] |         std::__insertion_sort() {
  1.549 us [196398] |             comp(1, 4) = 1;
  0.729 us [196398] |             memmove();
  0.252 us [196398] |             comp(3, 1) = 0;
  0.197 us [196398] |             comp(3, 4) = 1;
  0.219 us [196398] |             comp(3, 1) = 0;
  0.214 us [196398] |             comp(2, 1) = 0;
  0.186 us [196398] |             comp(2, 4) = 1;
  0.189 us [196398] |             comp(2, 3) = 1;
  0.184 us [196398] |             comp(2, 1) = 0;
  0.210 us [196398] |             comp(5, 1) = 0;
  0.230 us [196398] |             comp(5, 4) = 0;
  8.466 us [196398] |         } /* std::__insertion_sort */
10.011 us [196398] |     } /* test_std_sort */
10.568 us [196398] | } /* main */
```

std::sort (with functor)

```
#include <algorithm>
using namespace std;

struct Functor {
    bool operator()(int a, int b) {
        return a < b;
    }
};

__attribute__((noinline))
void test_std_sort_functor()
{
    int arr[] = { 4, 1, 3, 7, 2, 6, 5 };
    std::sort(begin(arr), end(arr), Functor());
}

int main()
{
    test_std_sort_functor();
}
```

std::sort (with functor)

```
$ uftrace std_sort_functor
# DURATION      TID      FUNCTION
  1.262 us [196476] | __monstartup();
  0.666 us [196476] | __cxa_atexit();
                     [196476] | main() {
                     [196476] |   test_std_sort_functor() {
  0.126 us [196476] |     std::__introsort_loop();
                     [196476] |     std::__insertion_sort() {
  1.120 us [196476] |       memmove();
  1.727 us [196476] |     } /* std::__insertion_sort */
  2.639 us [196476] |   } /* test_std_sort_functor */
  2.989 us [196476] | } /* main */
```

std::sort (with lambda)

```
#include <algorithm>
using namespace std;

__attribute__((noinline))
void test_std_sort_lambda()
{
    int arr[] = { 4, 1, 3, 7, 2, 6, 5 };
    std::sort(begin(arr), end(arr),
              [] (int a, int b) { return a < b; });
}

int main()
{
    test_std_sort_lambda();
}
```

std::sort (with lambda)

```
$ uftrace std_sort_lambda
# DURATION           TID           FUNCTION
  1.160  us [196525] | __monstartup();
  0.636  us [196525] | __cxa_atexit();
                      [196525] | main()  {
                      [196525] |   test_std_sort_lambda()  {
  0.181  us [196525] |     std::__introsort_loop();
                      [196525] |     std::__insertion_sort()  {
  0.897  us [196525] |       memmove();
  1.518  us [196525] |     } /* std::__insertion_sort */
  2.549  us [196525] |   } /* test_std_sort_lambda */
  2.894  us [196525] | } /* main */
```

실습

- sample 코드에서 sort 디렉터리 안에서 직접 테스트하면서 replay 결과를 비교해 보세요.

```
$ git clone https://github.com/honggyukim/ksc2019
```

```
$ cd ksc2019/sample
```

```
$ make
```

```
$ uftrace record -a sort_all
```

```
$ uftrace replay
```

```
$ uftrace graph
```

```
$ uftrace report
```

```
$ uftrace tui
```

STL Containers

Performance Comparison

std::vector

std::deque

std::list

Benchmark

Benchmark

```
std::vector<std::string> vec;

void bench_vector_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        vec.push_back(s);
}

int main()
{
    int iter = 3000000;
    bench_vector_push_back(iter);

}
```

Benchmark

```
std::vector<std::string> vec;  
  
void bench_vector_push_back(int iter) {  
    std::string s("Hello");  
    while (iter--)  
        vec.push_back(s);  
}
```

```
int main()  
{  
    int iter = 3000000;  
    bench_vector_push_back(iter);  
  
}
```

Benchmark

```
std::vector<std::string> vec;
std::deque<std::string> deq;

void bench_vector_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        vec.push_back(s);
}

void bench_deque_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        deq.push_back(s);
}

int main()
{
    int iter = 3000000;
    bench_vector_push_back(iter);
    bench_deque_push_back(iter);

}
```

Benchmark

```
std::vector<std::string> vec;
std::deque<std::string> deq;
std::list<std::string> lis;

void bench_vector_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        vec.push_back(s);
}

void bench_deque_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        deq.push_back(s);
}

void bench_list_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        lis.push_back(s);
}

int main()
{
    int iter = 3000000;
    bench_vector_push_back(iter);
    bench_deque_push_back(iter);
    bench_list_push_back(iter);
}
```

Benchmark

```
std::vector<std::string> vec;
std::deque<std::string> deq;
std::list<std::string> lis;

void bench_vector_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        vec.push_back(s);
}

void bench_deque_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        deq.push_back(s);
}

void bench_list_push_back(int iter) {
    std::string s("Hello");
    while (iter--)
        lis.push_back(s);
}

int main()
{
    int iter = 3000000;
    bench_vector_push_back(iter);
    bench_deque_push_back(iter);
    bench_list_push_back(iter);
}
```

```
$ uftrace record \
-d uftrace.data.bench \
--nest-libcall \
-A malloc@arg1 -R malloc@retval -A free@arg1 \
-A memcpy@arg3 -A memmove@arg3 \
./bench
```

```
$ uftrace record \
-d uftrace.data.bench \
--nest-libcall \
-A malloc@arg1 -R malloc@retval -A free@arg1 \
-A memcpy@arg3 -A memmove@arg3 \
./bench
```

```
$ uftrace graph
```

uftrace graph

- Show function call graph

uftrace graph

Record 된 데이터를 분석해서
함수 호출 그래프를 출력

```
$ uftrace graph
5.321  s : (1) main
2.176  s : +- (1) bench_vector_push_back
1.365  s : | +- (23) std::vector::_M_insert_aux
145.377 us : | | +- (23) operator new
122.596 us : | | | (23) malloc
:
685.339 ms : | | | +- (4194326) memcpy
:
2.888 ms : | | | +- (22) operator delete
2.857 ms : | | | (22) free
:
336.277 ms : | +- (2999977) memcpy
:
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : | +- (2812500) memcpy
:
167.695 ms : | +- (187500) std::deque::_M_push_back_aux
101.126 ms : | | +- (187515) operator new
60.892 ms : | | | (187515) malloc
:
14.972 ms : | | | +- (187500) memcpy
:
993.690 us : | | | +- (15) memmove
:
12.357 us : | | | +- (15) operator delete
5.924 us : | | | (15) free
:
2.418  s : +- (1) bench_list_push_back
1.057  s : | +- (3000000) operator new
423.438 ms : | | (3000000) malloc
:
230.213 ms : | | | +- (3000000) memcpy
:
199.812 ms : | | | +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph
5.321  s : (1) main
 2.176  s : +- (1) bench_vector_push_back
  1.365  s : |  +- (23) std::vector::_M_insert_aux
145.377 us : |  |  +- (23) operator new
122.596 us : |  |  |  (23) malloc
  :
  :
685.339 ms : |  |  |  +- (4194326) memcpy
  :
  :
2.888 ms : |  |  |  +- (22) operator delete
2.857 ms : |  |  |  (22) free
  :
  :
336.277 ms : |  |  +- (2999977) memcpy
  :
  :
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : |  +- (2812500) memcpy
  :
  :
167.695 ms : |  |  +- (187500) std::deque::_M_push_back_aux
101.126 ms : |  |  |  +- (187515) operator new
  60.892 ms : |  |  |  |  (187515) malloc
  :
  :
14.972 ms : |  |  |  +- (187500) memcpy
  :
  :
993.690 us : |  |  |  +- (15) memmove
  :
  :
12.357 us : |  |  |  +- (15) operator delete
  5.924 us : |  |  |  |  (15) free
  :
  :
  2.418  s : +- (1) bench_list_push_back
  1.057  s : |  +- (3000000) operator new
423.438 ms : |  |  (3000000) malloc
  :
  :
230.213 ms : |  |  +- (3000000) memcpy
  :
  :
199.812 ms : |  |  |  +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph
```

```
5.321  s : (1) main
 2.176  s : +- (1) bench_vector_push_back
  1.365  s : | +- (23) std::vector::_M_insert_aux
145.377 us : | | +- (23) operator new
122.596 us : | | | (23) malloc
  :
  :
685.339 ms : | | | +- (4194326) memcpy
  :
  :
2.888 ms : | | | +- (22) operator delete
2.857 ms : | | | (22) free
  :
  :
336.277 ms : | | +- (2999977) memcpy
  :
  :
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : | +- (2812500) memcpy
  :
  :
167.695 ms : | | +- (187500) std::deque::_M_push_back_aux
101.126 ms : | | | +- (187515) operator new
  60.892 ms : | | | (187515) malloc
  :
  :
14.972 ms : | | | +- (187500) memcpy
  :
  :
993.690 us : | | | +- (15) memmove
  :
  :
12.357 us : | | | +- (15) operator delete
  5.924 us : | | | (15) free
  :
  :
  2.418  s : +- (1) bench_list_push_back
  1.057  s : | +- (3000000) operator new
  423.438 ms : | | (3000000) malloc
  :
  :
230.213 ms : | | | +- (3000000) memcpy
  :
  :
199.812 ms : | | | +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph
```

```
5.321  s : (1) main
2.176  s : +- (1) bench_vector_push_back
1.365  s : | +- (23) std::vector::_M_insert_aux
145.377 us : | | +- (23) operator new
122.596 us : | | | (23) malloc
:
685.339 ms : | | +- (4194326) memcpy
:
2.888  ms : | | +- (22) operator delete
2.857  ms : | | | (22) free
:
336.277 ms : | +- (2999977) memcpy
:
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : | +- (2812500) memcpy
:
167.695 ms : | +- (187500) std::deque::_M_push_back_aux
101.126 ms : | | +- (187515) operator new
60.892  ms : | | | (187515) malloc
:
14.972  ms : | | +- (187500) memcpy
:
993.690  us : | | +- (15) memmove
:
12.357  us : | | +- (15) operator delete
5.924  us : | | | (15) free
:
2.418   s : +- (1) bench_list_push_back
1.057   s : | +- (3000000) operator new
423.438 ms : | | (3000000) malloc
:
230.213 ms : | | +- (3000000) memcpy
:
199.812 ms : | | +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph
```

```
5.321  s : (1) main
2.176  s : +- (1) bench_vector_push_back
1.365  s : |  +- (23) std::vector::_M_insert_aux
145.377 us : |  |  +- (23) operator new
122.596 us : |  |  |  (23) malloc
:
685.339 ms : |  |  |  +- (4194326) memcpy
:
2.888  ms : |  |  |  +- (22) operator delete
2.857  ms : |  |  |  (22) free
:
336.277 ms : |  |  +- (2999977) memcpy
:
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : |  +- (2812500) memcpy
:
167.695 ms : |  +- (187500) std::deque::_M_push_back_aux
101.126 ms : |  |  +- (187515) operator new
60.892  ms : |  |  |  (187515) malloc
:
14.972  ms : |  |  +- (187500) memcpy
:
993.690  us : |  |  +- (15) memmove
:
12.357  us : |  |  +- (15) operator delete
5.924  us : |  |  (15) free
:
2.418  s : +- (1) bench_list_push_back
1.057  s :   +- (3000000) operator new
423.438 ms :   |  (3000000) malloc
:
230.213 ms :   +- (3000000) memcpy
:
199.812 ms :   +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph

5.321  s : (1) main
2.176  s : +- (1) bench_vector_push_back
1.365  s : |  +- (23) std::vector::_M_insert_aux
145.377 us : |  |  +- (23) operator new
122.596 us : |  |  |  (23) malloc
        :
        :
685.339 ms : |  |  |  +- (4194326) memcpy
        :
        :
2.888 ms : |  |  |  +- (22) operator delete
2.857 ms : |  |  |  (22) free
        :
        :
336.277 ms : |  |  +- (2999977) memcpy
        :
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : |  +- (2812500) memcpy
        :
        :
167.695 ms : |  +- (187500) std::deque::_M_push_back_aux
101.126 ms : |  |  +- (187515) operator new
60.892 ms : |  |  |  (187515) malloc
        :
        :
14.972 ms : |  |  +- (187500) memcpy
        :
        :
993.690 us : |  |  +- (15) memmove
        :
        :
12.357 us : |  |  +- (15) operator delete
5.924 us : |  |  (15) free
        :
        :
2.418  s : +- (1) bench_list_push_back
1.057  s :  +- (3000000) operator new
423.438 ms : |  (3000000) malloc
        :
        :
230.213 ms :  +- (3000000) memcpy
        :
        :
199.812 ms :  +- (3000000) std::__detail::_List_node_base::_M_hook
```

```
$ uftrace graph
```

```
5.321  s : (1) main
2.176  s : +- (1) bench_vector_push_back
1.365  s : |  +- (23) std::vector::_M_insert_aux
145.377 us : |  |  +- (23) operator new
122.596 us : |  |  |  (23) malloc
:
:
685.339 ms : |  |  |  +- (4194326) memcpy
:
:
2.888 ms : |  |  |  +- (22) operator delete
2.857 ms : |  |  |  (22) free
:
:
336.277 ms : |  |  +- (2999977) memcpy
:
:
726.388 ms : +- (1) bench_deque_push_back
217.685 ms : |  +- (2812500) memcpy
:
:
167.695 ms : |  +- (187500) std::deque::_M_push_back_aux
101.126 ms : |  |  +- (187515) operator new
60.892 ms : |  |  |  (187515) malloc
:
:
14.972 ms : |  |  +- (187500) memcpy
:
:
993.690 us : |  |  +- (15) memmove
:
:
12.357 us : |  |  +- (15) operator delete
5.924 us : |  |  (15) free
:
:
2.418  s : +- (1) bench_list_push_back
1.057  s : |  +- (3000000) operator new
423.438 ms : |  |  (3000000) malloc
:
:
230.213 ms : |  |  +- (3000000) memcpy
:
:
199.812 ms : |  |  +- (3000000) std::__detail::_List_node_base::_M_hook
```

원본 버퍼의 내용을
새로운 버퍼로 복사하는데
필요한 **memcpy**

새로 들어온 **string** 을
push_back 해서
발생하는 **memcpy**

uftrace replay

시간 순서대로 출력해서 다시 확인

bench_vector_push_back

`std::vector<std::string>`

std::vector

```
v.push_back(1);
```



std::vector

```
v.push_back(2);
```



std::vector

```
v.push_back(3);
```



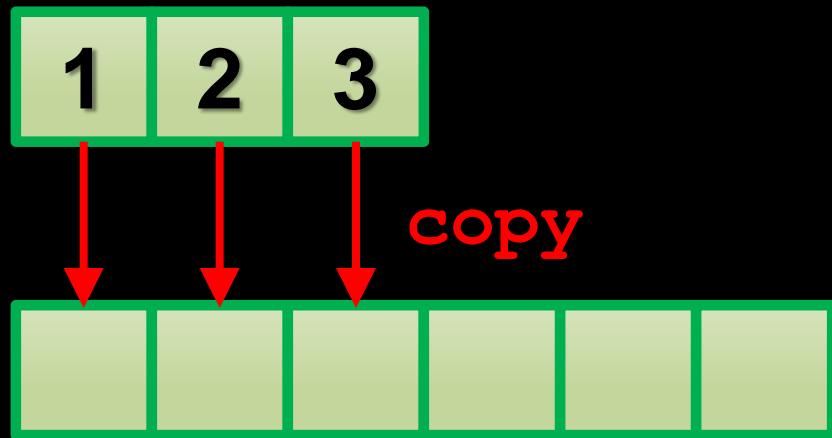
std::vector

```
v.push_back(4);
```



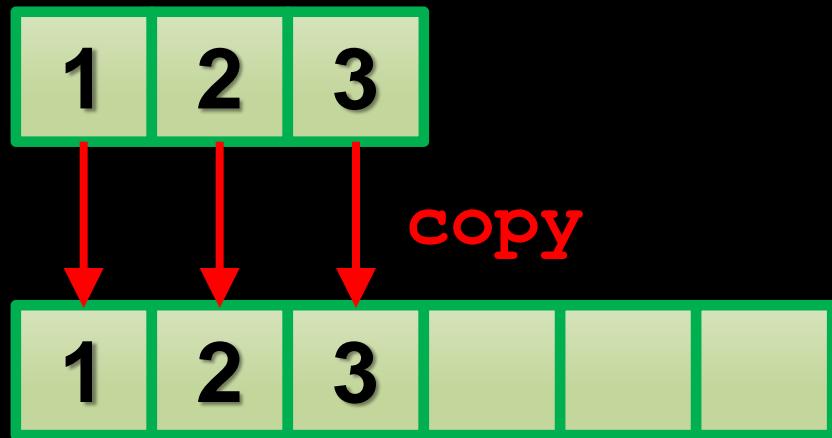
std::vector

v.push_back(4);



std::vector

v.push_back(4);



std::vector

```
v.push_back(4);
```



std::vector

```
v.push_back(4);
```



std::vector

```
v.push_back(5) ;
```



std::vector

```
v.push_back(6);
```



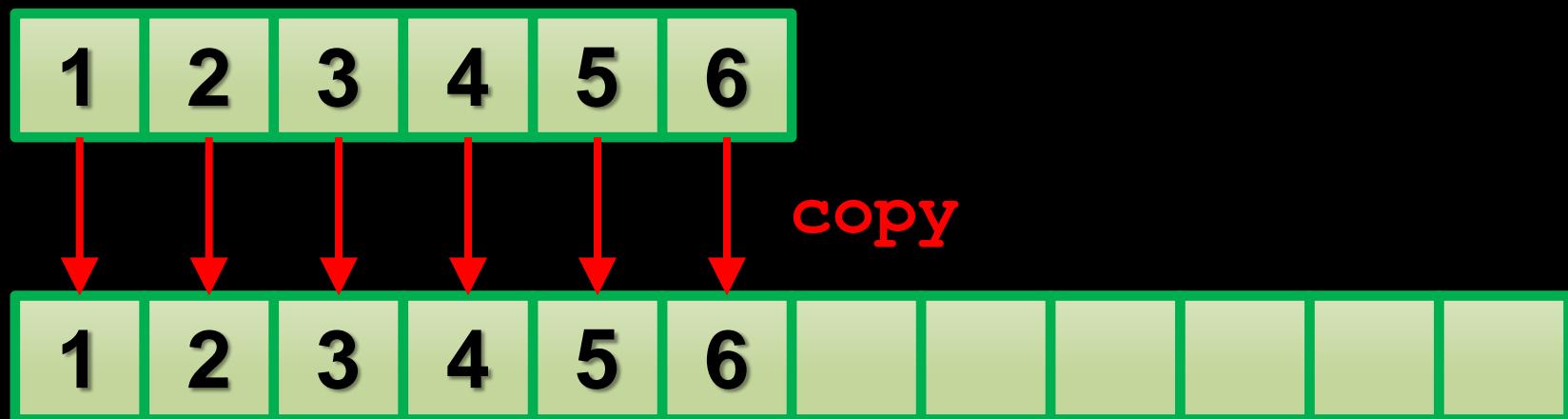
std::vector

```
v.push_back(7);
```



std::vector

v.push_back(7);



std::vector

```
v.push_back(7);
```



std::vector

```
v.push_back(7);
```



```
$ uftrace replay
```

```
    ...
[121878] | main() {
[121878] |   bench_vector_push_back() {
    ...
```

std::vector push_back

```
$ uftrace replay
```

```
    ...
[121878] | main()  {
[121878] |   bench_vector_push_back()  {
[121878] |     std::vector::_M_insert_aux()  {
[121878] |       operator new()  {
3.533 us [121878] |         malloc(32) = 0xdc6550;
4.200 us [121878] |       } /* operator new */
2.006 us [121878] |       memcpy(5);
7.777 us [121878] |     } /* std::vector::_M_insert_aux */
...

```

std::vector push_back

```
$ uftrace replay
```

```
    ...
[121878] | main() {
[121878] |   bench_vector_push_back() {
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
3.533 us [121878] |         malloc(32) = 0xdc6550;
4.200 us [121878] |       } /* operator new */
2.006 us [121878] |       memcpy(5);
7.777 us [121878] |     } /* std::vector::_M_insert_aux */
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
0.227 us [121878] |         malloc(64) = 0xdc6580;
0.780 us [121878] |       } /* operator new */
0.250 us [121878] |       memcpy(5);
0.160 us [121878] |       memcpy(5);
[121878] |       operator delete() {
1.813 us [121878] |         free(0xdc6550);
3.460 us [121878] |       } /* operator delete */
6.370 us [121878] |     } /* std::vector::_M_insert_aux */
...
}
```

std::vector push_back

```
$ uftrace replay
```

```
    ...
[121878] | main() {
[121878] |   bench_vector_push_back() {
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
3.533 us [121878] |         malloc(32) = 0xdc6550;
4.200 us [121878] |       } /* operator new */
2.006 us [121878] |       memcpy(5);
7.777 us [121878] |     } /* std::vector::_M_insert_aux */
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
0.227 us [121878] |         malloc(64) = 0xdc6580;
0.780 us [121878] |       } /* operator new */
0.250 us [121878] |       memcpy(5);
0.160 us [121878] |       memcpy(5);
[121878] |       operator delete() {
1.813 us [121878] |         free(0xdc6550);
3.460 us [121878] |       } /* operator delete */
6.370 us [121878] |     } /* std::vector::_M_insert_aux */
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
0.244 us [121878] |         malloc(128) = 0xdc65d0;
0.743 us [121878] |       } /* operator new */
0.186 us [121878] |       memcpy(5);
0.160 us [121878] |       memcpy(5);
0.160 us [121878] |       memcpy(5);
[121878] |       operator delete() {
0.320 us [121878] |         free(0xdc6580);
0.897 us [121878] |       } /* operator delete */
3.737 us [121878] |     } /* std::vector::_M_insert_aux */
...
}
```

std::vector push_back

```
$ uctrace replay
```

```
    ...
[121878] | main() {
[121878] |   bench_vector_push_back() {
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
[121878] |         malloc(32) = 0xdc6550;
[121878] |       } /* operator new */
[121878] |       memcpy(5);
[121878] |     } /* std::vector::_M_insert_aux */
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
[121878] |         malloc(64) = 0xdc6580;
[121878] |       } /* operator new */
[121878] |       memcpy(5);
[121878] |       memcpy(5);
[121878] |       operator delete() {
[121878] |         free(0xdc6550);
[121878] |       } /* operator delete */
[121878] |     } /* std::vector::_M_insert_aux */
[121878] |     std::vector::_M_insert_aux() {
[121878] |       operator new() {
[121878] |         malloc(128) = 0xdc65d0;
[121878] |       } /* operator new */
[121878] |       memcpy(5);
[121878] |       memcpy(5);
[121878] |       memcpy(5);
[121878] |       operator delete() {
[121878] |         free(0xdc6580);
[121878] |       } /* operator delete */
[121878] |     } /* std::vector::_M_insert_aux */
[121878] |     memcpy(5);
[121878] | ...
3.533 us [121878]
4.200 us [121878]
2.006 us [121878]
7.777 us [121878]
0.227 us [121878]
0.780 us [121878]
0.250 us [121878]
0.160 us [121878]
1.813 us [121878]
3.460 us [121878]
6.370 us [121878]
0.244 us [121878]
0.743 us [121878]
0.186 us [121878]
0.160 us [121878]
0.160 us [121878]
0.320 us [121878]
0.897 us [121878]
3.737 us [121878]
0.167 us [121878]
```

원본 버퍼의 내용을
새로운 버퍼로 복사하는데
필요한 **memcpy**

새로 들어온 string 을
push_back 해서
발생하는 **memcpy**

std::vector push_back

원본 버퍼의 내용을
새로운 버퍼로 복사하는데
필요한 `memcpy` 의
횟수가 증가함

std::vector push_back

```
[121878] |     ...
[121878] |     bench_vector_push_back()  {
[121878] |         ...
[121878] |         std::vector::_M_insert_aux()  {
[121878] |             operator new()  {
[1.040 us [121878] |                 malloc(1024) = 0xdc6980;
[1.510 us [121878] |             } /* operator new */
[0.157 us [121878] |             memcpy(5);
[0.157 us [121878] |             memcpy(5);
[0.156 us [121878] |             memcpy(5);
[0.157 us [121878] |             memcpy(5);
[0.154 us [121878] |             memcpy(5);
[0.153 us [121878] |             memcpy(5);
[0.150 us [121878] |             memcpy(5);
[0.150 us [121878] |             memcpy(5);
[0.157 us [121878] |             memcpy(5);
[0.153 us [121878] |             memcpy(5);
[0.153 us [121878] |             memcpy(5);
[0.146 us [121878] |             memcpy(5);
[0.150 us [121878] |             memcpy(5);
[0.154 us [121878] |             memcpy(5);
[0.157 us [121878] |             memcpy(5);
[0.156 us [121878] |             memcpy(5);
[121878] |             [121878] |             operator delete()  {
[0.274 us [121878] |                 free(0xdc6770);
[0.700 us [121878] |             } /* operator delete */
[9.369 us [121878] |         } /* std::vector::_M_insert_aux */
[0.157 us [121878] |         memcpy(5);
[0.170 us [121878] |         memcpy(5);
[0.160 us [121878] |         memcpy(5);
[121878] |     ...
[121878] |
```

원본 버퍼의 내용을
새로운 버퍼로 복사하는데
필요한 `memcpy` 의
횟수가 크게 증가함

새로 들어온 `string` 을
`push_back` 해서
발생하는 `memcpy` 는
일정한 횟수

`std::vector push_back`

bench_deque_push_back

`std::deque<std::string>`

std::deque

```
d.push_back(1);
```



std::deque

```
d.push_back(2);
```



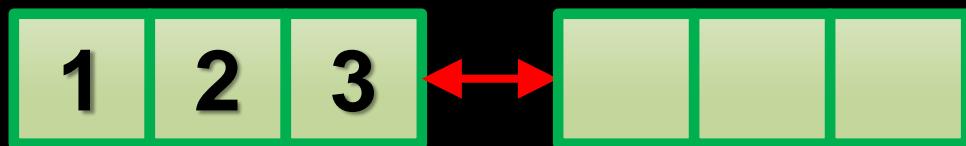
std::deque

d.push_back(3);



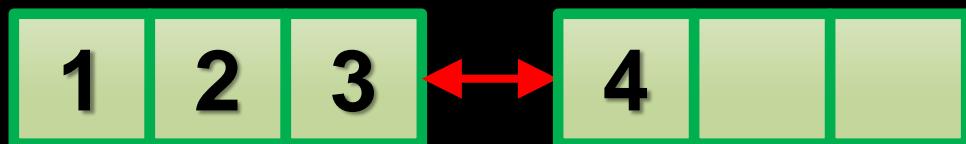
std::deque

`d.push_back(4);`



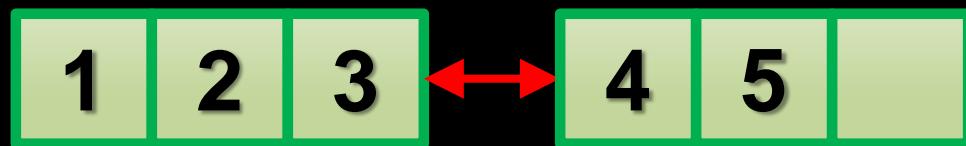
std::deque

d.push_back(4);



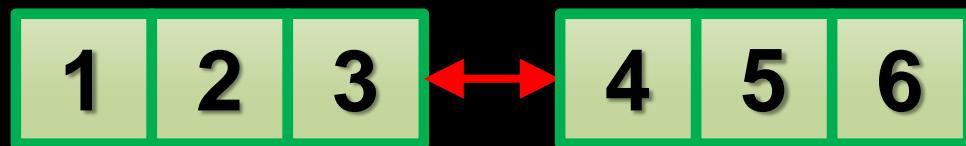
std::deque

d.push_back(5) ;



std::deque

d.push_back(6);



std::deque

d.push_back(7);



std::deque

d.push_back(7);



```
$ uftrace replay
```

```
...  
[121878] | bench_deque_push_back() {  
0.083 us [121878] |     memcpy(5);  
0.080 us [121878] |     memcpy(5);  
0.077 us [121878] |     memcpy(5);  
0.078 us [121878] |     memcpy(5);  
0.079 us [121878] |     memcpy(5);  
0.079 us [121878] |     memcpy(5);  
0.076 us [121878] |     memcpy(5);  
0.076 us [121878] |     memcpy(5);  
0.076 us [121878] |     memcpy(5);  
0.080 us [121878] |     memcpy(5);  
0.074 us [121878] |     memcpy(5);  
0.078 us [121878] |     memcpy(5);  
0.077 us [121878] |     memcpy(5);  
0.079 us [121878] |     memcpy(5);  
0.080 us [121878] |     memcpy(5);  
[121878] |     std::deque::_M_push_back_aux()  
[121878] |         operator new() {  
4.682 us [121878] |             malloc(512) = 0xdc6550;  
6.328 us [121878] |         } /* operator new */  
0.085 us [121878] |         memcpy(5)  
7.552 us [121878] |     } /* std::deque::_M_push_back_aux */  
0.074 us [121878] |     memcpy(5);  
0.080 us [121878] |     memcpy(5);  
0.080 us [121878] |     memcpy(5);  
0.083 us [121878] |     memcpy(5);  
0.077 us [121878] |     memcpy(5);  
...  
std::deque push_back
```

```
$ uftrace replay
```

```
...
[121878] | bench_deque_push_back() {
0.083 us [121878] |     memcpy(5);
0.080 us [121878] |     memcpy(5);
0.077 us [121878] |     memcpy(5);
0.078 us [121878] |     memcpy(5);
0.079 us [121878] |     memcpy(5);
0.079 us [121878] |     memcpy(5);
0.076 us [121878] |     memcpy(5);
0.076 us [121878] |     memcpy(5);
0.076 us [121878] |     memcpy(5);
0.080 us [121878] |     memcpy(5);
0.074 us [121878] |     memcpy(5);
0.078 us [121878] |     memcpy(5);
0.077 us [121878] |     memcpy(5);
0.079 us [121878] |     memcpy(5);
0.080 us [121878] |     memcpy(5);
[121878] |     std::deque::_M_push_back_aux() {
[121878] |         operator new() {
4.682 us [121878] |             malloc(512) = 0xdc6550;
6.328 us [121878] |         } /* operator new */
0.085 us [121878] |         memcpy(5);
7.552 us [121878] |     } /* std::deque::_M_push_back_aux */
0.074 us [121878] |     memcpy(5);
0.080 us [121878] |     memcpy(5);
0.080 us [121878] |     memcpy(5);
0.083 us [121878] |     memcpy(5);
0.077 us [121878] |     memcpy(5);
...
}
```

미리 할당된 chunk 버퍼에 push_back

15 번의 memcpy 호출:

32 bytes (std::string 의 크기) * 15
= 한 버퍼에서 480 바이트가 소모됨

추가적인
chunk
할당

std::deque push_back

```
$ uftrace replay
```

```
...  
[121878] |  bench_deque_push_back() {  
...  
[121878] |      std::deque::_M_push_back_aux() {  
[121878] |          operator new() {  
4.682 us [121878] |              malloc(512) = 0xdc6550;  
6.328 us [121878] |          } /* operator new */  
0.085 us [121878] |          memcpy(5);  
7.552 us [121878] |          } /* std::deque::_M_push_back_aux */  
0.074 us [121878] |          memcpy(5);  
0.080 us [121878] |          memcpy(5);  
0.080 us [121878] |          memcpy(5);  
0.083 us [121878] |          memcpy(5);  
0.077 us [121878] |          memcpy(5);  
0.079 us [121878] |          memcpy(5);  
0.075 us [121878] |          memcpy(5);  
0.078 us [121878] |          memcpy(5);  
0.075 us [121878] |          memcpy(5);  
0.077 us [121878] |          memcpy(5);  
0.077 us [121878] |          memcpy(5);  
0.077 us [121878] |          memcpy(5);  
0.079 us [121878] |          memcpy(5);  
0.080 us [121878] |          memcpy(5);  
0.078 us [121878] |          memcpy(5);  
[121878] |      std::deque::_M_push_back_aux() {  
[121878] |          operator new() {  
0.305 us [121878] |              malloc(512) = 0xdc6760;  
0.882 us [121878] |          } /* operator new */  
0.079 us [121878] |          memcpy(5);  
1.256 us [121878] |          } /* std::deque::_M_push_back_aux */
```

std::deque push_back

bench_list_push_back

std::list<std::string>

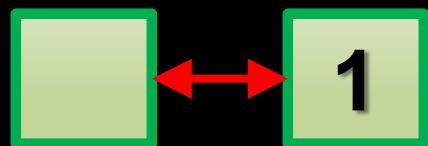
std::list

```
l.push_back(1);
```



std::list

```
l.push_back(1);
```



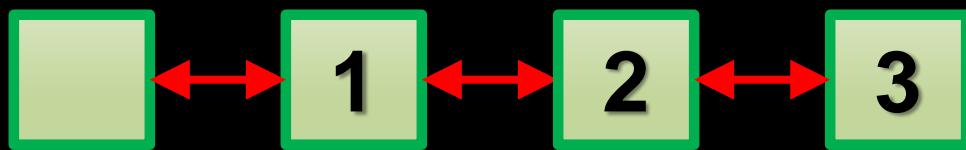
std::list

```
l.push_back(2);
```



std::list

l.push_back(3);



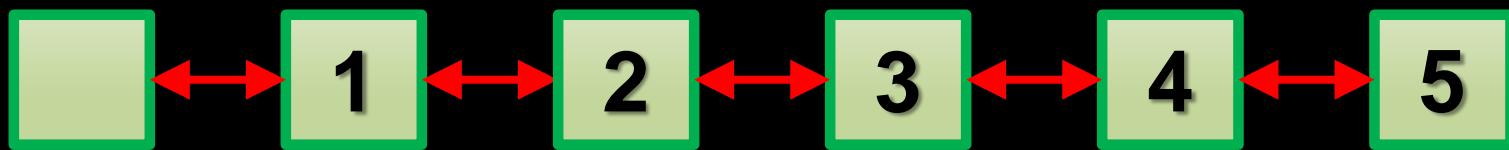
std::list

l.push_back(4);



std::list

l.push_back(5);



```
$ uftrace replay
```

```
...
[121878] |    bench_list_push_back() {
[121878] |        operator new() {
0.450 us [121878] |            malloc(48) = 0xdc62f0;      32 bytes of std::string
0.677 us [121878] |        } /* operator new */
0.080 us [121878] |        memcpy(5);
5.139 us [121878] |        std::__detail::_List_node_base::_M_hook();
[121878] |        operator new() {
0.240 us [121878] |            malloc(48) = 0xfd2580;      + 8 bytes of pointer * 2
0.480 us [121878] |        } /* operator new */
0.083 us [121878] |        memcpy(5);
0.080 us [121878] |        std::__detail::_List_node_base::_M_hook();
[121878] |        operator new() {
0.400 us [121878] |            malloc(48) = 0xdca0a0;
0.641 us [121878] |        } /* operator new */
0.085 us [121878] |        memcpy(5);
0.071 us [121878] |        std::__detail::_List_node_base::_M_hook();
[121878] |        operator new() {
0.251 us [121878] |            malloc(48) = 0x1cac6d0;
0.479 us [121878] |        } /* operator new */
0.075 us [121878] |        memcpy(5);
0.069 us [121878] |        std::__detail::_List_node_base::_M_hook();
[121878] |        operator new() {
0.304 us [121878] |            malloc(48) = 0xdc8200;
0.511 us [121878] |        } /* operator new */
0.076 us [121878] |        memcpy(5);
0.066 us [121878] |        std::__detail::_List_node_base::_M_hook();
...

```

**32 bytes of std::string
+ 8 bytes of pointer * 2**

size of "Hello"

std::list push_back

```
$ uftrace graph bench_list_push_back
#
# function graph for 'bench_list_push_back' (session: 53a12394b0ce1367)
#
backtrace
=====
backtrace #0: hit 1, time 2.418 s
[0] main (0x400cc9)
[1] bench_list_push_back (0x401044)

calling functions
=====
2.418 s : (1) bench_list_push_back
1.057 s : +- (3000000) operator new
423.438 ms : | (3000000) malloc
             :
230.213 ms : +- (3000000) memcpy
             :
199.812 ms : +- (3000000) std::__detail::_List_node_base::__M_hook
```

std::list push_back

Clang / LLVM

Analyzing Clang

```
$ uctrace -t 2ms -F ccl_main ./clang fibonacci.c
# DURATION      TID      FUNCTION
              [ 9045] | ccl_main() {
              [ 9045] |   clang::CompilerInvocation::CreateFromArgs() {
2.270 ms  [ 9045] |     ParseCodeGenArgs();
8.653 ms  [ 9045] |   } /* clang::CompilerInvocation::CreateFromArgs */
              [ 9045] |   clang::ExecuteCompilerInvocation() {
              [ 9045] |     clang::CompilerInstance::ExecuteAction() {
2.185 ms  [ 9045] |       clang::FrontendAction::BeginSourceFile();
              [ 9045] |       clang::FrontendAction::Execute() {
              [ 9045] |         clang::CodeGenAction::ExecuteAction() {
              [ 9045] |           clang::ASTFrontendAction::ExecuteAction() {
              [ 9045] |             clang::ParseAST() {
              [ 9045] |               clang::Parser::Initialize() {
3.841 ms  [ 9045] |                 clang::Preprocessor::Lex();
3.887 ms  [ 9045] |               } /* clang::Parser::Initialize */
              [ 9045] |               clang::BackendConsumer::HandleTranslationUnit() {
              [ 9045] |                 clang::EmitBackendOutput() {
              [ 9045] |                   llvm::LLVMTargetMachine::addPassesToEmitFile() {
2.044 ms  [ 9045] |                     addPassesToGenerateCode();
2.068 ms  [ 9045] |                   } /* llvm::LLVMTargetMachine::addPassesToEmitFile */
              [ 9045] |                     llvm::legacy::PassManager::run() {
2.196 ms  [ 9045] |                       llvm::legacy::PassManagerImpl::run();
2.196 ms  [ 9045] |                     } /* llvm::legacy::PassManager::run */
5.053 ms  [ 9045] |                   } /* clang::EmitBackendOutput */
5.076 ms  [ 9045] |                 } /* clang::BackendConsumer::HandleTranslationUnit */
23.361 ms [ 9045] |               } /* clang::ParseAST */
23.385 ms [ 9045] |             } /* clang::ASTFrontendAction::ExecuteAction */
23.385 ms [ 9045] |               } /* clang::CodeGenAction::ExecuteAction */
23.386 ms [ 9045] |             } /* clang::FrontendAction::Execute */
25.651 ms [ 9045] |           } /* clang::CompilerInstance::ExecuteAction */
25.667 ms [ 9045] |         } /* clang::ExecuteCompilerInvocation */
34.368 ms [ 9045] |       } /* ccl_main */
```

Analyzing Clang

```
$ uctrace -t 2ms -F ccl_main ./clang fibonacci.c
# DURATION      TID      FUNCTION
              [ 9045] |  ccl_main() {
              [ 9045] |    clang::CompilerInvocation::CreateFromArgs() {
2.270 ms   [ 9045] |      ParseCodeGenArgs();
8.653 ms   [ 9045] |    } /* clang::CompilerInvocation::CreateFromArgs */
              [ 9045] |    clang::ExecuteCompilerInvocation() {
              [ 9045] |      clang::CompilerInstance::ExecuteAction() {
2.185 ms   [ 9045] |        clang::FrontendAction::BeginSourceFile();
              [ 9045] |        clang::FrontendAction::Execute() {
              [ 9045] |          clang::CodeGenAction::ExecuteAction() {
              [ 9045] |          clang::ASTFrontendAction::ExecuteAction() {
              [ 9045] |            clang::ParseAST() {
              [ 9045] |              clang::Parser::Initialize() {
3.841 ms   [ 9045] |                clang::Preprocessor::Lex();
3.887 ms   [ 9045] |              } /* clang::Parser::Initialize */
              [ 9045] |              clang::BackendConsumer::HandleTranslationUnit() {
              [ 9045] |                clang::EmitBackendOutput() {
              [ 9045] |                  llvm::LLVMTargetMachine::addPassesToEmitFile() {
2.044 ms   [ 9045] |                    addPassesToGenerateCode();
2.068 ms   [ 9045] |                  } /* llvm::LLVMTargetMachine::addPassesToEmitFile */
              [ 9045] |                  llvm::legacy::PassManager::run() {
2.196 ms   [ 9045] |                    llvm::legacy::PassManagerImpl::run();
2.196 ms   [ 9045] |                  } /* llvm::legacy::PassManager::run */
5.053 ms   [ 9045] |                  } /* clang::EmitBackendOutput */
5.076 ms   [ 9045] |                  } /* clang::BackendConsumer::HandleTranslationUnit */
23.361 ms  [ 9045] |                } /* clang::ParseAST */
              [ 9045] |              } /* clang::ASTFrontendAction::ExecuteAction */
23.385 ms  [ 9045] |              } /* clang::CodeGenAction::ExecuteAction */
23.386 ms  [ 9045] |            } /* clang::FrontendAction::Execute */
25.651 ms  [ 9045] |            } /* clang::CompilerInstance::ExecuteAction */
25.667 ms  [ 9045] |          } /* clang::ExecuteCompilerInvocation */
34.368 ms  [ 9045] |        } /* ccl_main */
```

ParseAST

Analyzing Clang

```
$ uctrace -t 2ms -F ccl_main ./clang fibonacci.c
# DURATION      TID      FUNCTION
              [ 9045] | ccl_main() {
              [ 9045] |   clang::CompilerInvocation::CreateFromArgs() {
2.270 ms  [ 9045] |     ParseCodeGenArgs();
8.653 ms  [ 9045] |   } /* clang::CompilerInvocation::CreateFromArgs */
              [ 9045] |   clang::ExecuteCompilerInvocation() {
              [ 9045] |     clang::CompilerInstance::ExecuteAction() {
2.185 ms  [ 9045] |       clang::FrontendAction::BeginSourceFile();
              [ 9045] |       clang::FrontendAction::Execute() {
              [ 9045] |         clang::CodeGenAction::ExecuteAction() {
              [ 9045] |           clang::ASTFrontendAction::ExecuteAction() {
              [ 9045] |             clang::ParseAST() {
              [ 9045] |               clang::Parser::Initialize() {
              [ 9045] |                 clang::Preprocessor::Lex();
3.841 ms  [ 9045] |               } /* clang::Parser::Initialize */      Backend Code Gen
3.887 ms  [ 9045] |               clang::BackendConsumer::HandleTranslationUnit() {
              [ 9045] |                 clang::EmitBackendOutput() {
              [ 9045] |                   llvm::LLVMTargetMachine::addPassesToEmitFile() {
              [ 9045] |                     addPassesToGenerateCode();
2.044 ms  [ 9045] |                   } /* llvm::LLVMTargetMachine::addPassesToEmitFile */
2.068 ms  [ 9045] |                     llvm::legacy::PassManager::run() {
              [ 9045] |                       llvm::legacy::PassManagerImpl::run();
              [ 9045] |                     } /* llvm::legacy::PassManager::run */
5.053 ms  [ 9045] |                   } /* clang::EmitBackendOutput */
5.076 ms  [ 9045] |                 } /* clang::BackendConsumer::HandleTranslationUnit */
23.361 ms [ 9045] |               } /* clang::ParseAST */
23.385 ms [ 9045] |             } /* clang::ASTFrontendAction::ExecuteAction */
23.385 ms [ 9045] |               } /* clang::CodeGenAction::ExecuteAction */
23.386 ms [ 9045] |             } /* clang::FrontendAction::Execute */
25.651 ms [ 9045] |           } /* clang::CompilerInstance::ExecuteAction */
25.667 ms [ 9045] |         } /* clang::ExecuteCompilerInvocation */
34.368 ms [ 9045] |       } /* ccl_main */
```

Analyzing Clang TMP expansion

```
$ clang++ tmpfib.cc
```

```
#include <iostream>

#define fibnum 8
template <unsigned N> struct Fibonacci {
    enum { value = Fibonacci<N-1>::value + Fibonacci<N-2>::value };
};
template <> struct Fibonacci<1> { enum { value = 1 }; };
template <> struct Fibonacci<0> { enum { value = 0 }; };

int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci<fibnum>::value;
    std::cout << std::endl;
}
```

Analyzing Clang TMP expansion

```
$ clang++ tmpfib.cc
```

```
#include <iostream>

#define fibnum 8
template <unsigned N> struct Fibonacci {
    enum { value = Fibonacci<N-1>::value + Fibonacci<N-2>::value };
};
template <> struct Fibonacci<1> { enum { value = 1 }; };
template <> struct Fibonacci<0> { enum { value = 0 }; };

int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci<fibnum>::value;
    std::cout << std::endl;
}
```

Analyzing Clang TMP expansion

```
$ clang++ tmpfib.cc
```

```
#include <iostream>

#define fibnum 8
template <unsigned N> struct Fibonacci {
    enum { value = Fibonacci<N-1>::value + Fibonacci<N-2>::value };
};

template <> struct Fibonacci<1> { enum { value = 1 }; };
template <> struct Fibonacci<0> { enum { value = 0 }; };

int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci<fibnum>::value;
    std::cout << std::endl;
}
```

Recursive Expansion

Analyzing Clang TMP expansion

```
$ clang++ tmpfib.cc
```

```
#include <iostream>

#define fibnum 8
template <unsigned N> struct Fibonacci {
    enum { value = Fibonacci<N-1>::value + Fibonacci<N-2>::value };
};

template <> struct Fibonacci<1> { enum { value = 1 }; };
template <> struct Fibonacci<0> { enum { value = 0 }; };

int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci<fibnum>::value;
    std::cout << std::endl;
}
```

Recursive Expansion

Analyzing Clang TMP expansion

```
$ uftrace record -t 1ms clang++ tmpfib.cc
```

```
#include <iostream>

#define fibnum 8
template <unsigned N> struct Fibonacci {
    enum { value = Fibonacci<N-1>::value + Fibonacci<N-2>::value };
};

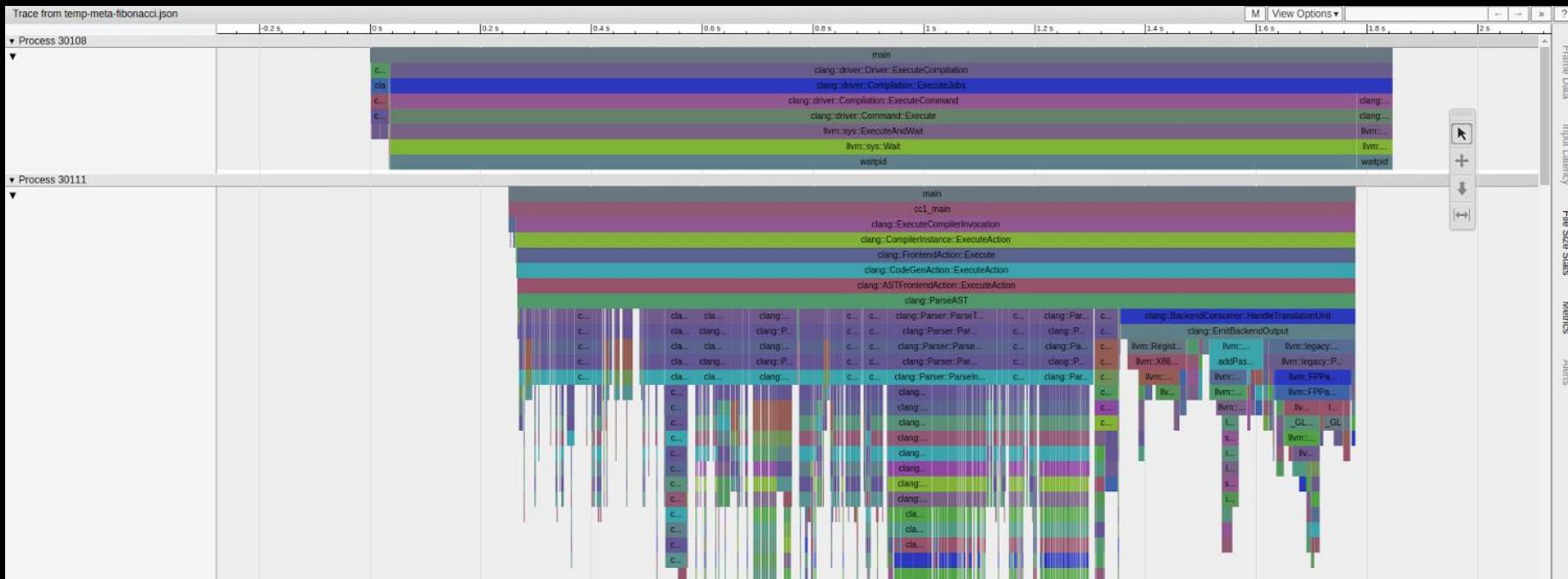
template <> struct Fibonacci<1> { enum { value = 1 }; };
template <> struct Fibonacci<0> { enum { value = 0 }; };

int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci<fibnum>::value;
    std::cout << std::endl;
}
```

Recursive Expansion

Analyzing Clang TMP expansion

- Clang TMP expansion에 대한 chrome trace 결과
 - <https://uftrace.github.io/dump/clang.tmp.fib.html>



Analyzing Clang TMP expansion

```
$ uftrace record -t 1ms clang++ constexpr-fib.cc
```

```
#include <iostream>
```

C++14 **constexpr** function

```
#define fibnum 8
constexpr int Fibonacci(const int n)
{
    if (n <= 2)
        return 1;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
int main(void)
{
    std::cout << "Fibonacci(" << fibnum << ") = ";
    std::cout << Fibonacci(fibnum);
    std::cout << std::endl;
}
```

uftrace tui

Text User Interface

TUI: Text User Interface

```
$ man uftrace tui
```

UFTRACE-TUI(1)

UFTRACE-TUI(1)

NAME

`uftrace-tui` - (Interactive) Text-based User Interface

SYNOPSIS

`uftrace tui [options]`

DESCRIPTION

This command starts an interactive window on a terminal which can show same output of other commands like `graph`, `report` and `info`. Users can navigate the result easily with key presses. The command line options are used to limit the initial data loading.

OPTIONS

`-F FUNC`, `--filter=FUNC`

Set filter to trace selected functions only. This option can be used more than once. See `uftrace-replay(1)` for an explanation of filters.

...

TUI: Text User Interface

```
$ uftrace record -a ./clang hello.c  
$ uftrace tui -t 1ms
```

Key uftrace command

```
G call Graph for session #1: clang-6.0
call Graph for session #2: clang-6.0
call Graph for session #3: x86_64-linux-gnu-ld.bfd
R Report functions
I uftrace Info
h Help message
q quit
```

Key uftrace command

```
G call Graph for session #1: clang-6.0
call Graph for session #2: clang-6.0
call Graph for session #3: x86_64-linux-gnu-ld.bfd
R Report functions
I uftrace Info
h Help message
q quit
```

```
Help: (press any key to exit)

ARROW          Navigation
PgUp/Dn
Home/End
Enter          Select/Fold
G              Show (full) call graph
g              Show call graph for this function
R              Show uftrace report
I              Show uftrace info
S              Change session
O              Open editor
c/e            Collapse/Expand graph
n/p            Next/Prev sibling
u              Move up to parent
l              Move to the longest executed child
j/k            Move down/up
/              Search
</>/N/P        Search next/prev
v              Show debug message
h/?            Show this help
q              Quit
```

Key uftrace command

G call Graph for session #1: clang-6.0
call Graph for session #2: clang-6.0
call Graph for session #3: x86_64-linux-gnu-ld.bfd
R Report functions
I uftrace Info
h Help message
q quit

```
uftrace info
# system information
# =====
# program version      : v0.8.3-326-g480f0
# recorded on          : Thu Aug  2 14:52:13 2018
# cmdline              : uftrace record -a clang /home/honggyu/hello.c
# cpu info              : Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
# number of cpus        : 40 / 40 (online / possible)
# memory info           : 9.6 / 62.5 GB (free / total)
# system load            : 9.35 / 11.69 / 9.99 (1 / 5 / 15 min)
# kernel version         : Linux 4.4.0-116-generic
# hostname              : intel20
# distro                : "Ubuntu 16.04.3 LTS"
#
# process information
# =====
# number of tasks        : 3
# task list              : 153294(clang), 153315(clang-6.0), 153324(ld)
# exe image              : /home/honggyu/work/llvm/release/build.release.g.pg/bin/clang-6.0
# auto-args               : true
# pattern                : regex
# exit status              : exited with code: 0
# elapsed time            : 36.550426950 sec
# cpu time                : 0.636 / 35.836 sec (sys / user)
# context switch           : 16 / 55 (voluntary / involuntary)
# max rss                  : 1325216 KB
# page fault              : 3 / 239746 (major / minor)
# disk iops                 : 264 / 19448 (read / write)
```

Key uftrace command

G call Graph for session #1: clang-6.0
call Graph for session #2: clang-6.0
call Graph for session #3: x86_64-linux-gnu-ld.bfd

R Report functions

I uftrace Info

h Help message

q quit

Total Time	Self Time	Calls	Function
37.986 s	29.594 us	3	llvm::raw_ostream::operator<<
19.623 s	3.287 ms	2	main
18.993 s	217.431 us	1	clang::driver::Driver::ExecuteCompilation
18.993 s	21.904 us	2	llvm::yaml::Scanner::scanTag
18.993 s	957.012 us	2	llvm::IRBuilderBase::CreateBinaryIntrinsic
18.992 s	15.747 us	2	llvm::sys::Wait
18.992 s	18.992 s	2	waitpid
567.307 ms	292.219 us	1	ccl_main
552.319 ms	84.215 us	1	clang::CodeGen::CodeGenFunction::EmitOMPSimdFinal
552.235 ms	459.279 us	1	clang::CompilerInstance::ExecuteAction
537.654 ms	3.276 us	1	clang::FrontendAction::Execute
537.651 ms	0.814 us	1	clang::CodeGenAction::ExecuteAction
537.650 ms	211.023 us	1	clang::ASTFrontendAction::ExecuteAction
537.439 ms	179.817 ms	1	clang::ParseAST
203.407 ms	111.046 us	100	clang::Parser::ParseTopLevelDecl
203.296 ms	153.751 us	102	llvm::WriteThinLinkBitcodeToFile
157.805 ms	68.595 ms	76	llvm::ValueEnumerator::OptimizeConstants
140.190 ms	140.190 ms	1	xexit
131.668 ms	291.452 us	79	clang::Parser::ParseDeclarationOrFunctionDefinition
131.377 ms	19.252 ms	79	clang::Parser::ParseDeclOrFunctionDefInternal
128.702 ms	103.632 ms	28	clang::Preprocessor::Lex
75.820 ms	22.581 us	22	clang::Parser::ExpectAndConsumeSemi
73.558 ms	295.771 us	1	clang::BackendConsumer::HandleTranslationUnit
73.262 ms	3.986 ms	1	clang::EmitBackendOutput
71.474 ms	63.913 us	21	clang::Parser::ParseDeclaration
71.410 ms	5.187 ms	21	std::__introsort_loop
50.500 ms	486.983 us	1	llvm::WriteBitcodeToFile
48.032 ms	0.953 us	1	llvm::legacy::PassManager::run
48.031 ms	539.554 us	1	std::vector::_M_range_insert
43.342 ms	8.545 us	2	llvm::sys::fs::createTemporaryFile
41.484 ms	388.087 us	2	llvm::FPPassManager::runOnModule
41.096 ms	6.050 ms	3	llvm::FPPassManager::runOnFunction
34.868 ms	155.527 us	3	clang::driver::createDriverOptTable
32.407 ms	225.644 us	1	clang::driver::Driver::BuildCompilation
27.129 ms	84.861 us	3	clang::Preprocessor::EnterSourceFile
27.044 ms	6.210 us	3	clang::SrcMgr::ContentCache::getBuffer
27.038 ms	83.267 us	3	clang::FileManager::getBufferForFile
26.955 ms	5.154 us	3	_GLOBAL__N_1::RealFile::getBuffer
26.950 ms	5.760 us	3	llvm::MemoryBuffer::getOpenFile
26.944 ms	34.826 us	3	llvm::DICompileUnit::getEmissionKind
26.909 ms	26.909 ms	3	pread
26.742 ms	38.983 us	1	std::vector::_M_realloc_insert
26.703 ms	60.510 us	1	llvm::X86TargetMachine::getSubtargetImpl
26.642 ms	411.204 us	1	llvm::X86Subtarget::X86Subtarget
25.317 ms	14.310 us	1	llvm::sys::path::stem
25.302 ms	35.562 us	2	llvm::sys::Process::GetArgumentVector
25.284 ms	270.026 us	2	llvm::sys::ProcessInfo::ProcessInfo
25.059 ms	4.691 us	6	clang::Lexer::Lex

Key uftrace command

```
G call Graph for session #1: clang-6.0
call Graph for session #2: clang-6.0
call Graph for session #3: x86_64-linux-gnu-ld.bfd
R Report functions
I uftrace Info
h Help message
q quit
```

TOTAL TIME : FUNCTION

```
570.475 ms : (1) clang-6.0
570.475 ms : (1) main
567.307 ms : (1) ccl_main
14.695 ms : (1) clang::CompilerInvocation::CreateFromArgs
11.412 ms : (1) clang::driver::createDriverOptTable
3.349 ms : (1) llvm::opt::OptTable::OptTable
:
7.999 ms : (2) llvm::opt::OptTable::addValues
:
2.226 ms : (1) llvm::opt::OptTable::ParseArgs
1.237 ms : (1) llvm::opt::OptTable::ParseOneArg
:
552.319 ms : (1) clang::CodeGen::CodeGenFunction::EmitOMPSimdFinal
552.235 ms : (1) clang::CompilerInstance::ExecuteAction
14.121 ms : (1) clang::DiagnosticConsumer::~DiagnosticConsumer
3.127 ms : (1) llvm::InnerLoopVectorizer::widenInstruction
2.381 ms : (1) clang::vfs::recursive_directory_iterator::recursive_directory_iterator
2.346 ms : (1) clang::vfs::File::~File
:
10.437 ms : (1) clang::Builtin::Context::initializeBuiltins
:
537.654 ms : (1) clang::FrontendAction::Execute
537.651 ms : (1) clang::CodeGenAction::ExecuteAction
537.650 ms : (1) clang::ASTFrontendAction::ExecuteAction
537.439 ms : (1) clang::ParseAST
14.959 ms : (1) clang::Preprocessor::EnterMainSourceFile
14.907 ms : (1) clang::Preprocessor::EnterSourceFile
14.870 ms : (1) clang::SrcMgr::ContentCache::getBuffer
14.867 ms : (1) FileManager::getBufferForFile
14.832 ms : (1) _GLOBAL__N_1::RealFile::getBuffer
14.830 ms : (1) llvm::MemoryBuffer::getOpenFile
14.828 ms : (1) llvm::DIBuildUnit::getEmissionKind
14.816 ms : (1) pread
:
50.500 ms : (1) llvm::WriteBitcodeToFile
50.013 ms : (1) clang::Preprocessor::Lex
9.395 ms : (1) clang::Lexer::Lex
9.394 ms : (1) clang::ento::registerStreamChecker
9.394 ms : (1) clang::Preprocessor::HandleDirective
9.384 ms : (1) clang::ento::registerCStringSyntaxChecker
9.157 ms : (1) clang::Preprocessor::EnterSourceFile
9.133 ms : (1) clang::SrcMgr::ContentCache::getBuffer
9.131 ms : (1) clang::FileManager::getBufferForFile
9.107 ms : (1) _GLOBAL__N_1::RealFile::getBuffer
9.105 ms : (1) llvm::MemoryBuffer::getOpenFile
9.103 ms : (1) llvm::DIBuildUnit::getEmissionKind
9.092 ms : (1) pread
:
```

TOTAL TIME : FUNCTION

```
570.475 ms : (1) clang-6.0
570.475 ms : (1) main
567.307 ms : (1) ccl_main
14.695 ms :   (1) clang::CompilerInvocation::CreateFromArgs
11.412 ms :     (1) clang::driver::createDriverOptTable
3.349 ms :       (1) llvm::opt::OptTable
:
7.999 ms :         (2) llvm::opt::OptTable::addValues
:
2.226 ms :           (1) llvm::opt::OptTable::ParseArgs
1.237 ms :             (1) llvm::opt::OptTable::ParseOneArg
:
552.319 ms :               (1) clang Help: (press any key to exit)
552.235 ms :               (1) clang
14.121 ms :                 (1) cl ARROW Navigation
3.127 ms :                   (1) PgUp/Dn
2.381 ms :                   (1) Home/End
2.346 ms :                   (1) Enter Select/Fold
:
10.437 ms :                     (1) G Show (full) call graph
537.654 ms :                     (1) g Show call graph for this function
537.651 ms :                     (1) R Show uctrace report
537.650 ms :                     (1) I Show uctrace info
537.439 ms :                     (1) S Change session
537.439 ms :                     (1) O Open editor
537.439 ms :                     (1) c/e Collapse/Expand graph
14.959 ms :                       (1) cl n/p Next/Prev sibling
14.907 ms :                       (1) u Move up to parent
14.870 ms :                       (1) l Move to the longest executed child
14.867 ms :                       (1) j/k Move down/up
14.832 ms :                       (1) /
14.830 ms :                       (1) Search
14.828 ms :                       (1) </>/N/P Search next/prev
14.828 ms :                       (1) v Show debug message
14.816 ms :                       (1) h/? Show this help
50.500 ms :                         (1) q Quit
50.013 ms :               (1)
9.395 ms :                 (1) clang::Lexer::Lex
9.394 ms :                 (1) clang::ento::registerStreamChecker
9.394 ms :                 (1) clang::Preprocessor::HandleDirective
9.384 ms :                 (1) clang::ento::registerCStringSyntaxChecker
9.157 ms :                 (1) clang::Preprocessor::EnterSourceFile
9.133 ms :                 (1) clang::SrcMgr::ContentCache::getBuffer
9.131 ms :                 (1) clang::FileManager::getBufferForFile
9.107 ms :                 (1) _GLOBAL__N_1::RealFile::getBuffer
9.105 ms :                 (1) llvm::MemoryBuffer::getOpenFile
9.103 ms :                 (1) llvm::DIBuildUnit::getEmissionKind
9.092 ms :                 (1) pread
```

TOTAL TIME : FUNCTION

```
570.475 ms : (1) clang-6.0
570.475 ms : (1) main
567.307 ms : (1) ccl_main
14.695 ms :    ► (1) clang::CompilerInvocation::CreateFromArgs
:
552.319 ms :    (1) clang::CodeGen::CodeGenFunction::EmitOMPSimdFinal
552.235 ms :    (1) clang::CompilerInstance::ExecuteAction
14.121 ms :    ► (1) clang::DiagnosticConsumer::~DiagnosticConsumer
:
537.654 ms :    (1) clang::FrontendAction::Execute
537.651 ms :    (1) clang::CodeGenAction::ExecuteAction
537.650 ms :    (1) clang::ASTFrontendAction::ExecuteAction
537.439 ms :    (1) clang::ParseAST
14.959 ms :    ► (1) clang::Preprocessor::EnterMainSourceFile
:
50.500 ms :    ► (1) llvm::WriteBitcodeToFile
:
2.586 ms :    ► (1) llvm::BitcodeWriter::writeThinLinkBitcode
:
200.824 ms :    (99) clang::Parser::ParseTopLevelDecl
200.715 ms :    (99) llvm::WriteThinLinkBitcodeToFile
68.905 ms :    ► (20) clang::Parser::ParseDeclaration
:
128.474 ms :    (77) clang::Parser::ParseDeclarationOrFunctionDefinition
128.190 ms :    (77) clang::Parser::ParseDeclOrFunctionDefInternal
8.857 ms :    ► (3) llvm::initializeWriteBitcodePassPass
:
97.288 ms :    (58) llvm::ValueEnumerator::OptimizeConstants
21.122 ms :    (7) clang::Parser::ExpectAndConsumeSemi
21.115 ms :    (7) clang::Preprocessor::Lex
:
3.777 ms :    ► (2) llvm::ValueEnumerator::print
:
9.612 ms :    ► (2) std::__Rb_tree::__M_emplace_hint_unique
:
2.891 ms :    ► (1) clang::Preprocessor::Lex
:
3.195 ms :    ► (2) llvm::WriteThinLinkBitcodeToFile
:
15.192 ms :    ► (2) clang::driver::tools::amdgpu::getAMDGPUTargetFeatures
:
73.558 ms :    ► (1) clang::BackendConsumer::HandleTranslationUnit
:
2.178 ms :    ► (1) llvm::llvm_shutdown
```

V8 JavaScript Engine

TOTAL TIME : FUNCTION

```
865.436 ms : (1) d8
834.139 ms :   |-(138) sched-in
               :
31.297 ms :     |-(1) v8::internal::GCExtension::GC
31.290 ms :       |-(1) v8::Isolate::RequestGarbageCollectionForTesting
31.287 ms :       |-(1) v8::internal::Heap::CollectAllGarbage
31.285 ms :       |-(1) v8::internal::Heap::CollectGarbage
40.573 us :         |-(1) v8::internal::GCTracer::Start
               :
16.968 us :           |-(1) v8::internal::Heap::GarbageCollectionPrologue
               :
30.659 ms :             |-(1) v8::internal::Heap::PerformGarbageCollection
150.236 us :               |-(1) v8::internal::SemiSpace::Commit
               :
679.669 us :               |-(1) v8::internal::StoreBuffer::MoveAllEntriesToRememberedSet
               :
29.626 ms :                 |-(1) v8::internal::Heap::MarkCompact
30.383 us :                   |-(1) v8::internal::PauseAllocationObserversScope::PauseAllocationObserversScope
               :
35.649 us :                   |-(1) v8::internal::PagedSpace::SetReadAndWritable
               :
67.791 us :                   |-(1) v8::internal::MarkCompactCollector::Prepare
               :
30.692 us :                     |-(1) v8::internal::Heap::MarkCompactPrologue
               :
28.685 ms :                       |-(1) v8::internal::MarkCompactCollector::CollectGarbage
22.614 ms :                         |-(1) v8::internal::MarkCompactCollector::MarkLiveObjects
               :
135.463 us :                         |-(1) v8::internal::MarkCompactCollector::ClearNonLiveReferences
               :
0.508 us :                           |-(1) v8::internal::MarkCompactCollector::VerifyMarking
               :
0.249 us :                           |-(1) v8::internal::MarkCompactCollector::RecordObjectStats
               :
76.772 us :                           |-(1) v8::internal::MarkCompactCollector::StartSweepSpaces
               :
5.673 ms :                             |-(1) v8::internal::MarkCompactCollector::Evacuate
               :
178.967 us :                             |-(1) v8::internal::MarkCompactCollector::Finish
```

```
28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage
22.614 ms :   (1) v8::internal::MarkCompactCollector::MarkLiveObjects
30.403 us :     (10) v8::internal::GCTracer::Scope::Scope
24.390 us :       (10) v8::internal::Heap::MonotonicallyIncreasingTimeInMs
1.143 us :         (10) v8::internal::V8::GetCurrentPlatform
:
12.364 us :           (10) v8::platform::DefaultPlatform::MonotonicallyIncreasingTime
7.132 us :             (10) v8::base::TimeTicks::Now
1.566 us :               (10) clock_gettime
:
16.998 us :                 (10) v8::internal::tracing::TraceEventHelper::GetTracingController
1.246 us :                   (10) v8::internal::V8::GetCurrentPlatform
:
2.104 us :                     (10) v8::internal::wasm::WasmInterpreter::Thread::NumInterpretedCalls
:
5.720 us :                       (10) v8::platform::tracing::TracingController::GetCategoryGroupEnabled
:
4.450 us :                         (1) v8::internal::StackGuard::PushInterruptsScope
1.444 us :                           (1) v8::base::Mutex::Lock
0.370 us :                             (1) pthread_mutex_lock
:
0.220 us :                               (1) v8::internal::Heap::SetStackLimits
:
0.950 us :                                 (1) v8::base::Mutex::Unlock
0.274 us :                                   (1) pthread_mutex_unlock
:
44.078 us :                                     (10) v8::internal::GCTracer::Scope::~Scope
36.678 us :                                       (10) v8::internal::Heap::MonotonicallyIncreasingTimeInMs
1.837 us :                                         (10) v8::internal::V8::GetCurrentPlatform
:
17.960 us :                                           (10) v8::platform::DefaultPlatform::MonotonicallyIncreasingTime
9.936 us :                                             (10) v8::base::TimeTicks::Now
1.939 us :                                               (10) clock_gettime
:
0.234 us :                                                 (1) v8::internal::LocalEmbedderHeapTracer::EnterFinalPause
```

```
28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage
22.614 ms :   ▶(1) v8::internal::MarkCompactCollector::MarkLiveObjects
:
135.463 us :   ▶(1) v8::internal::MarkCompactCollector::ClearNonLiveReferences
:
0.508 us :   -(1) v8::internal::MarkCompactCollector::VerifyMarking
:
0.249 us :   -(1) v8::internal::MarkCompactCollector::RecordObjectStats
:
76.772 us :   ▶(1) v8::internal::MarkCompactCollector::StartSweepSpaces
:
5.673 ms :   ▶(1) v8::internal::MarkCompactCollector::Evacuate
:
178.967 us :   ▶(1) v8::internal::MarkCompactCollector::Finish
```

```
28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage
22.614 ms :   ▶(1) v8::internal::MarkCompactCollector::MarkLiveObjects
:
135.463 us :   ▶(1) v8::internal::MarkCompactCollector::ClearNonLiveReferences
:
0.508 us :   -(1) v8::internal::MarkCompactCollector::VerifyMarking
:
0.249 us :   -(1) v8::internal::MarkCompactCollector::RecordObjectStats
:
76.772 us :   ▶(1) v8::internal::MarkCompactCollector::StartSweepSpaces
:
5.673 ms :   ▶(1) v8::internal::MarkCompactCollector::Evacuate
:
178.967 us :   ▶(1) v8::internal::MarkCompactCollector::Finish
```

```
Help: (press any key to exit)
```

ARROW	Navigation
PgUp/Dn	
Home/End	
Enter	Select/Fold
G	Show (full) call graph
g	Show call graph for this function
R	Show utrace report
I	Show utrace info
S	Change session
O	Open editor
c/e	Collapse/Expand graph
n/p	Next/Prev sibling
u	Move up to parent
l	Move to the longest executed child
j/k	Move down/up
/	Search
</>/N/P	Search next/prev
v	Show debug message
h/?	Show this help
q	Quit

'c': Collapse graph

CollectGarbage

```
28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage
22.614 ms :    ► (1) v8::internal::MarkCompactCollector::MarkLiveObjects
               :
135.463 us :    ► (1) v8::internal::MarkCompactCollector::ClearNonLiveReferences
               :
               :
0.508 us :    (1) v8::internal::MarkCompactCollector::VerifyMarking
               :
               :
0.249 us :    (1) v8::internal::MarkCompactCollector::RecordObjectStats
               :
               :
76.772 us :    ► (1) v8::internal::MarkCompactCollector::StartSweepSpaces
               :
               :
5.673 ms :    ► (1) v8::internal::MarkCompactCollector::Evacuate
               :
               :
178.967 us :    ► (1) v8::internal::MarkCompactCollector::Finish
```

```
$ uftrace record --auto-args -E linux:schedule ./d8 --expose-gc -e 'gc()'

$ uftrace tui -F v8::internal::GCExtension::GC -t 10us \
             -T v8::internal::MarkCompactCollector::CollectGarbage@time=0s
```

CollectGarbage

```
void MarkCompactCollector::CollectGarbage() {
    // Make sure that Prepare() has been called. The individual steps below will
    // update the state as they proceed.
    DCHECK(state_ == PREPARE_GC);

#ifndef ENABLE_MINOR_MC
    heap()->minor_mark_compact_collector()->CleanupSweepToIteratePages();
#endif // ENABLE_MINOR_MC

    MarkLiveObjects();
    ClearNonLiveReferences();
    VerifyMarking();

    RecordObjectStats();

    StartSweepSpaces();

    Evacuate();

    Finish();
}
```

CollectGarbage

```
void MarkCompactCollector::CollectGarbage() {
    // Make sure that Prepare() has been called. The individual steps below will
    // update the state as they proceed.
    DCHECK(state_ == PREPARE_GC);

#ifndef ENABLE_MINOR_MC
    heap()->minor_mark_compact_collector()->CleanupSweepToIteratePages();
#endif // ENABLE_MINOR_MC

    MarkLiveObjects();
    ClearNonLiveReferences();
    VerifyMarking();

    RecordObjectStats();

    StartSweepSpaces();      28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage
                            22.614 ms : ▶ (1) v8::internal::MarkCompactCollector::MarkLiveObjects
Evacuate();           135.463 us : ▶ (1) v8::internal::MarkCompactCollector::ClearNonLiveReferences
                      :
Finish();            0.508 us : (1) v8::internal::MarkCompactCollector::VerifyMarking
                      :
                      0.249 us : (1) v8::internal::MarkCompactCollector::RecordObjectStats
                      :
                      76.772 us : ▶ (1) v8::internal::MarkCompactCollector::StartSweepSpaces
                      :
                      5.673 ms : ▶ (1) v8::internal::MarkCompactCollector::Evacuate
                      :
                      178.967 us : ▶ (1) v8::internal::MarkCompactCollector::Finish
```

CollectGarbage

```
void MarkCompactCollector::CollectGarbage() {  
    // Make sure that Prepare() has been called. The individual steps below will  
    // update the state as they proceed.  
    DCHECK(state_ == PREPARE_GC);
```

실행되지 않는 코드

```
#ifdef ENABLE_MINOR_MC  
    heap()->minor_mark_compact_collector()->CleanupSweepToIteratePages();  
#endif // ENABLE_MINOR_MC
```

```
MarkLiveObjects();  
ClearNonLiveReferences();  
VerifyMarking();  
  
RecordObjectStats();
```

```
StartSweepSpaces();      28.685 ms : (1) v8::internal::MarkCompactCollector::CollectGarbage  
                           22.614 ms : ▶ (1) v8::internal::MarkCompactCollector::MarkLiveObjects  
Evacuate();             135.463 us : ▶ (1) v8::internal::MarkCompactCollector::ClearNonLiveReferences  
                           :  
Finish();               0.508 us : (1) v8::internal::MarkCompactCollector::VerifyMarking  
                           :  
                           0.249 us : (1) v8::internal::MarkCompactCollector::RecordObjectStats  
                           :  
                           76.772 us : ▶ (1) v8::internal::MarkCompactCollector::StartSweepSpaces  
                           :  
                           5.673 ms : ▶ (1) v8::internal::MarkCompactCollector::Evacuate  
                           :  
                           178.967 us : ▶ (1) v8::internal::MarkCompactCollector::Finish
```

(Python)
Scripting Support

```
$ gcc -pg test.c
$ uftrace -S count.py a.out
3
```

FUNCTION

```
main() {
    foo() {
        bar() {
            } /* bar */
        } /* foo */
    } /* main */
```

```
$ cat count.py
count = 0

def uftrace_begin():
    pass

def uftrace_entry(args):
    global count
    count += 1

def uftrace_exit(args):
    pass

def uftrace_end():
    print(count)
```

Context Info to Script

```
/* context information passed to script */
script_context = {
    int          tid;
    int          depth;
    long         timestamp;
    long         duration;      # exit only
    long         address;
    string       name;
    list         args;          # entry only (if available)
    value        retval;        # exit only (if available)
};
```

man pages

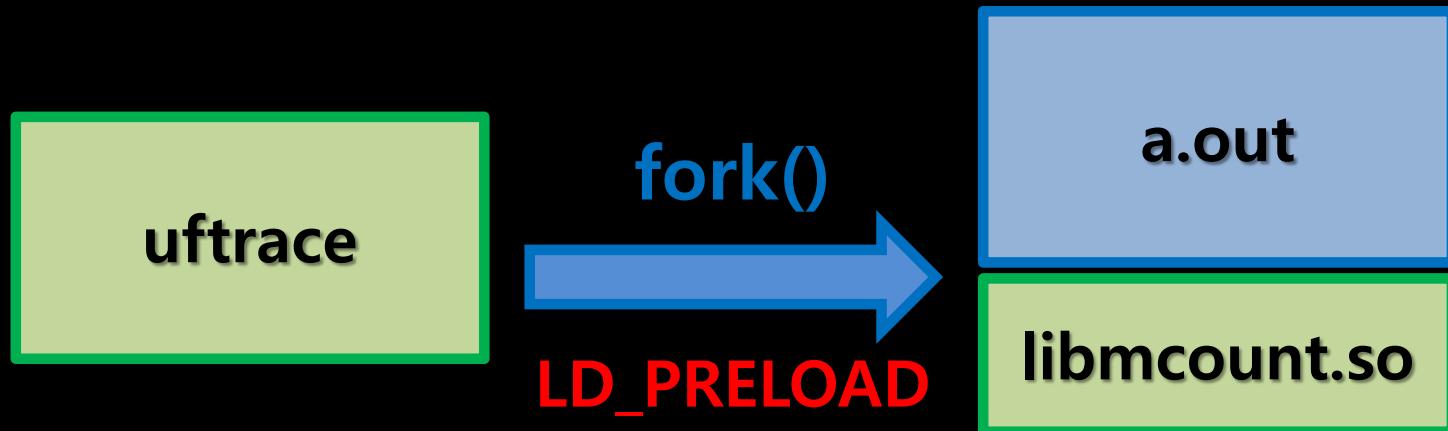
- **uftrace** 의 세부 옵션 설명은 manual 페이지에서 참고 가능
 - man uftrace
 - man uftrace record
 - man uftrace replay
 - man uftrace live
 - man uftrace report
 - man uftrace dump
 - man uftrace graph
 - man uftrace script
 - man uftrace tui
 - man uftrace info
 - man uftrace recv

uftrace Internals

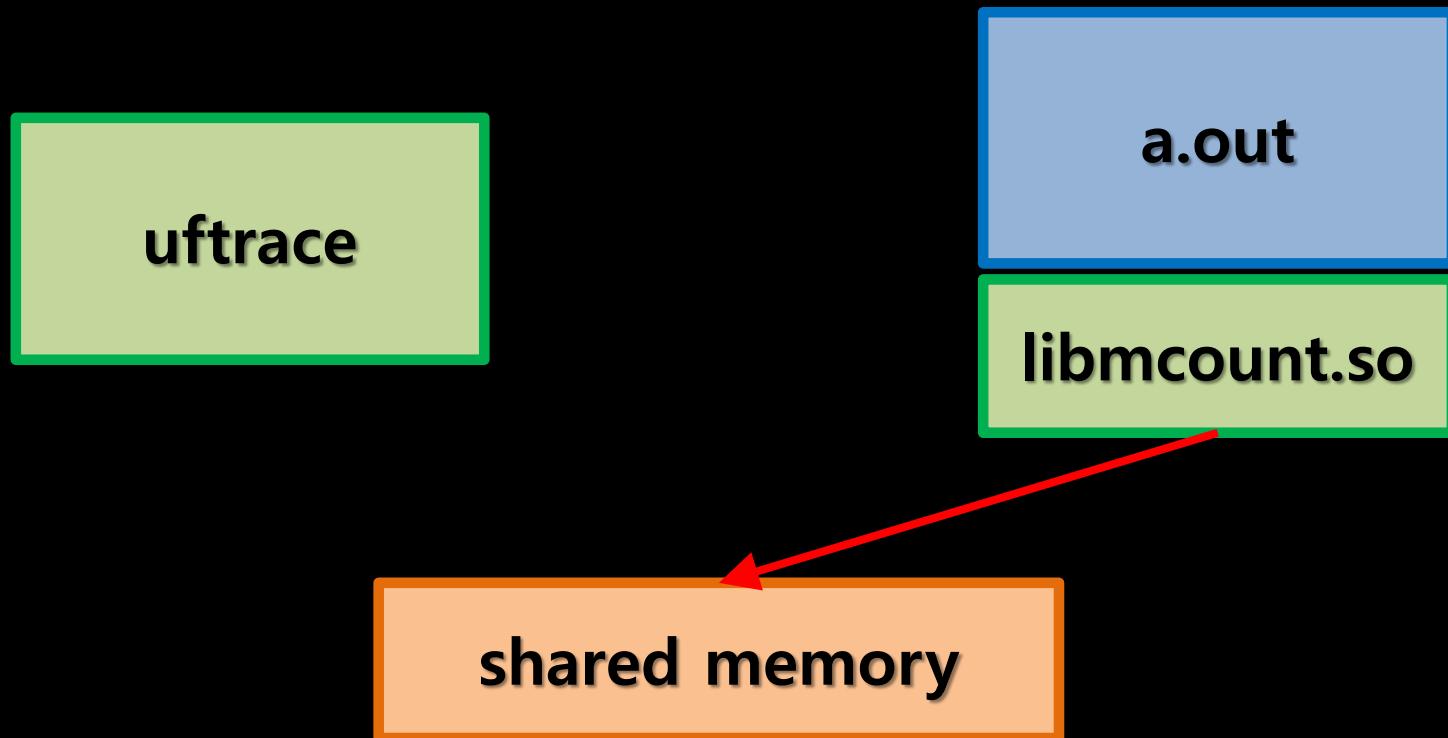
uftrace Internal Overview

uftrace

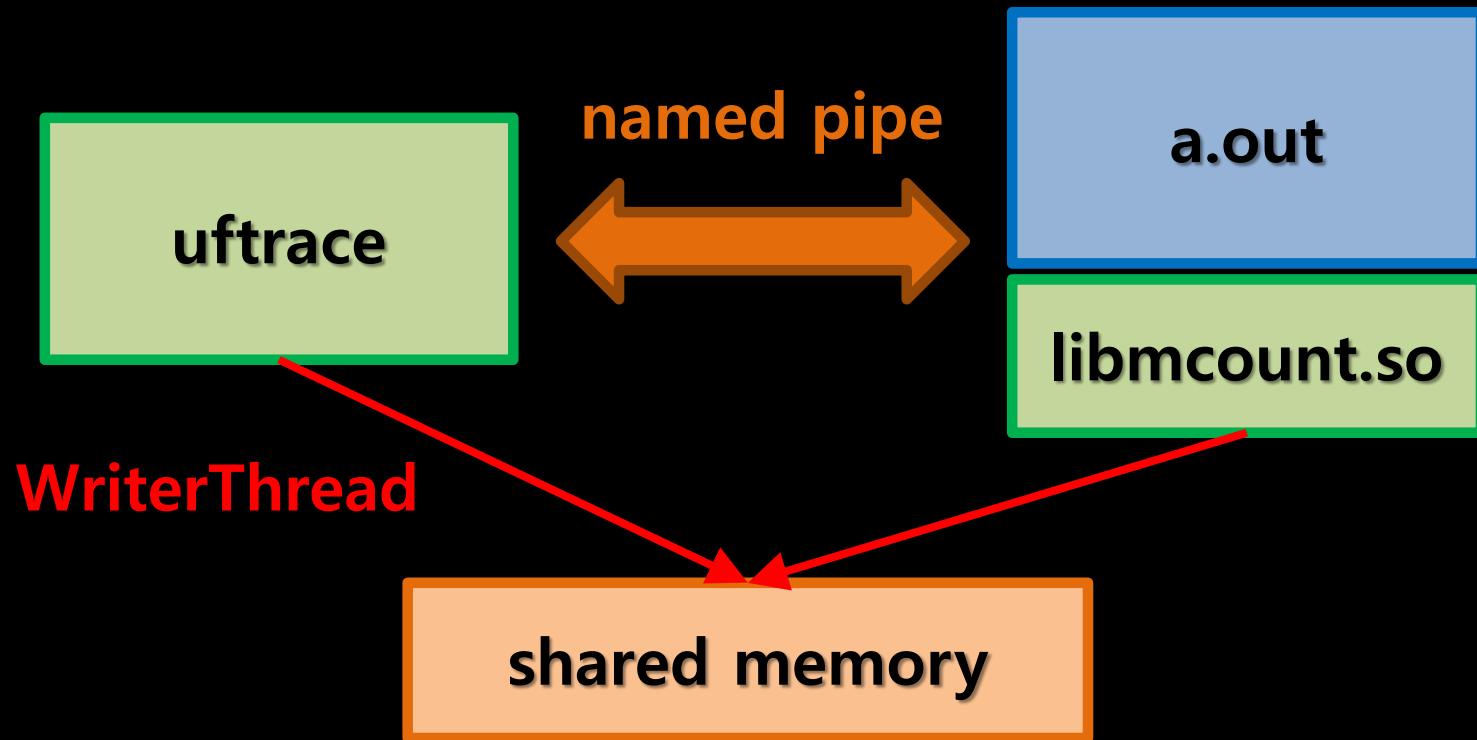
uftrace Internal Overview



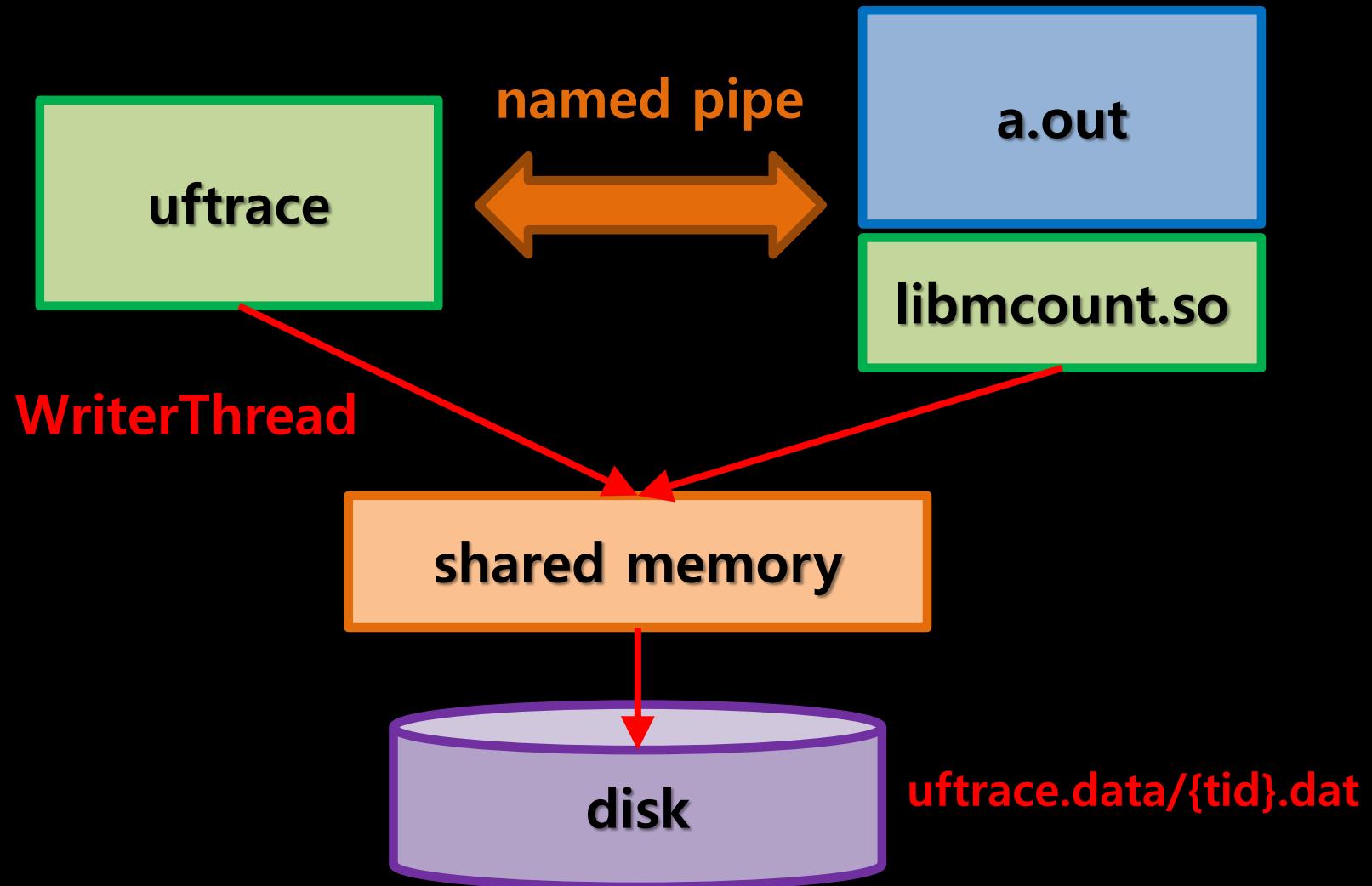
uftrace Internal Overview



uftrace Internal Overview



uftrace Internal Overview



uftrace Task Graph

- uftrace 가 record 를 수행하는 동작의 task graph 정보

```
$ uftrace graph --task
=====
# TOTAL TIME      SELF TIME      TID      TASK NAME
404.929 ms    321.692 ms    [ 4230] : uftrace*
                                         : |
278.662 us    278.662 us    [ 4241] : +---t-abc*
                                         : |
33.754 ms      4.061 ms     [ 4242] : +-WriterThread
27.415 ms      120.992 us    [ 4244] : +-WriterThread
27.212 ms      8.119 ms     [ 4245] : +-WriterThread
26.754 ms      6.616 ms     [ 4248] : +-WriterThread
26.859 ms      8.154 ms     [ 4247] : +-WriterThread
26.509 ms      1.645 ms     [ 4243] : +-WriterThread
25.320 ms      57.350 us    [ 4246] : +-WriterThread
24.757 ms      4.391 ms     [ 4249] : +-WriterThread
26.040 ms      3.707 ms     [ 4250] : +-WriterThread
24.004 ms      3.999 ms     [ 4251] : +-WriterThread
```

mcount_thread_data

- 각 thread마다 별도로 갖는 최상위 정보를 관리
 - TLS(Thread Local Storage) 변수로 생성

```
struct mcount_thread_data {
    int tid;
    int idx;

    ...

    struct mcount_ret_stack rstack[1024];
    void argbuf[1024];

    ...

    struct mcount_shmem shmem;

    ...

};

extern TLS struct mcount_thread_data mtd;
```

mcount_ret_stack

- 함수 진입과 반환시에 다양한 정보를 기록하며 관리

```
struct mcount_ret_stack {
    unsigned long *parent_loc; /* 실제 return 주소값의 주소 */
    unsigned long parent_ip;   /* 실제 return 주소 */
    unsigned long child_ip;   /* mcount 다음 주소 */
    enum mcount_rstack_flag flags;
    /* time in nsec (CLOCK_MONOTONIC) */
    uint64_t start_time;     /* 함수 진입(entry) 시간 */
    uint64_t end_time;       /* 함수 반환(exit) 시간 */
    int tid;
    ...
    struct plthook_data *pd;
    /* set arg_spec at entry and use it at exit */
    struct list_head *pargs;
};
```

mcount_shmem

- Tracing 정보가 기록되는 shared memory 정보를 관리
 - 여러개의 버퍼로 관리

```
struct mcount_shmem {  
    unsigned  
    int  
    int  
    int  
    int  
    bool  
    struct mcount_shmem_buffer  
};  
  
struct mcount_shmem_buffer {  
    unsigned size;  
    unsigned flag;  
    unsigned unused[2];  
    char data[];  
}
```

uftrace 내부 분석 자료

- **uftrace live 과정을 record 한 data**
 - <https://uftrace.github.io/data/uftrace-live.data.tar.gz>
- **uftrace 자체를 record 한 chrome trace image**
 - <https://uftrace.github.io/dump/uftrace-live.html>

```
$ gcc hello.c  
$ ./a.out
```

\$ gcc hello.c

\$./a.out

a.out

```
$ gcc hello.c
```

```
$ ./a.out
```

a.out



```
printf("Hello, World!")
```

```
$ gcc -pg hello.c
```

```
$ gcc -pg hello.c  
$ ./a.out
```

```
$ gcc -pg hello.c
```

```
$ ./a.out
```

libc.so

a.out

```
$ gcc -pg hello.c
```

```
$ ./a.out
```

libc.so

a.out



```
$ gcc -pg hello.c
```

```
$ ./a.out
```

libc.so

a.out

mcount()



```
$ gcc -pg hello.c  
$ ./a.out
```

libc.so

a.out

mcount()

함수 진입 기록

gmon.out
생성



```
$ gcc -pg hello.c  
$ ./a.out
```

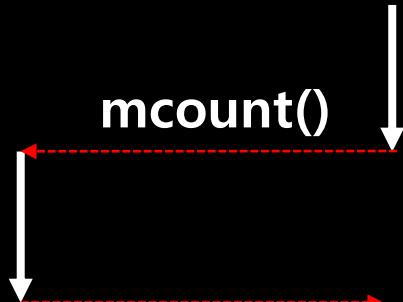
libc.so

a.out

mcount()

함수 진입 기록

gmon.out
생성



```
$ gcc -pg hello.c  
$ ./a.out
```

libc.so

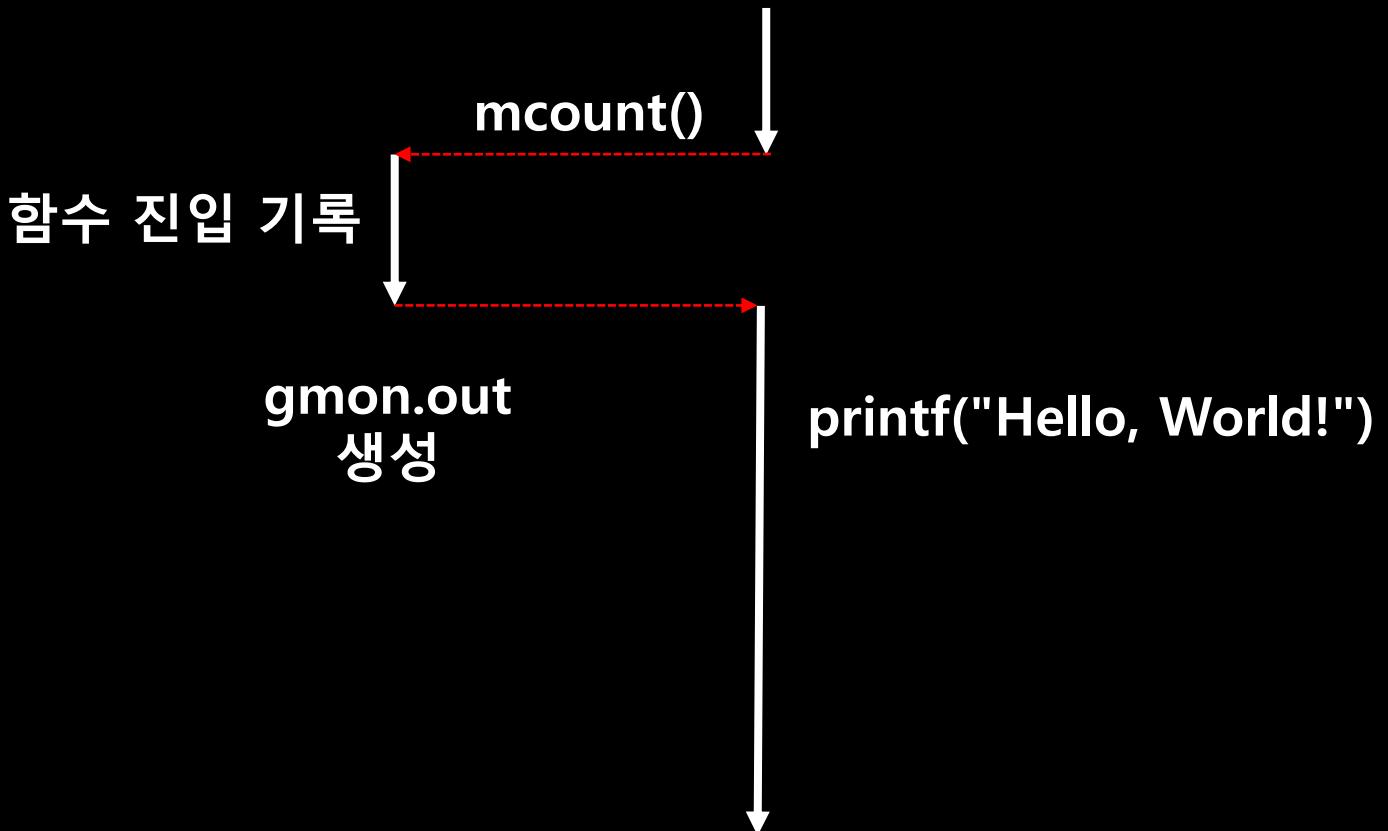
a.out

mcount()

함수 진입 기록

gmon.out
생성

printf("Hello, World!")



```
$ gcc -pg hello.c  
$ ./a.out
```

libc.so

a.out

mcount()

함수 진입 기록

gmon.out
생성

printf("Hello, World!")

gprof 툴로 프로파일
데이터 분석

```
$ uftrace ./a.out
```

```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```

```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```

mcount()
↓



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



원래 함수의 반환 주소를 별도의 공간에 보관해두고
실제 반환 주소를 `libmcount.so`로 변경해 함수가 리턴될때
실행 흐름을 얻을 수 있도록 함.

```
static int __mcount_entry(unsigned long *parent_loc, unsigned long child,
                         struct mcount_regs *regs)
{
    ...
    /* fixup the parent_loc in an arch-dependant way (if needed) */
    parent_loc = mcount_arch_parent_location(&syms, parent_loc, child);

    rstack = &mtdp->rstack[mtdp->idx++];

    rstack->depth      = mtdp->record_idx;
    rstack->dyn_idx    = MCOUNT_INVALID_DYNIDX;
    rstack->parent_loc = parent_loc;
    rstack->parent_ip   = *parent_loc;
    rstack->child_ip    = child;
    rstack->start_time  = mcount_gettime();
    rstack->end_time    = 0;
    rstack->flags       = 0;
    rstack->nr_events   = 0;
    rstack->event_idx   = ARGBUF_SIZE; return 주소를 저장하는 공간의 값을 변경
```

```
/* hijack the return address of child */
*parent_loc = mcount_return_fn; /* 일반적으로 mcount_return */
```

```
/* restore return address of parent */
if (mcount_auto_recover)
    mcount_auto_restore(mtdp);
...
```

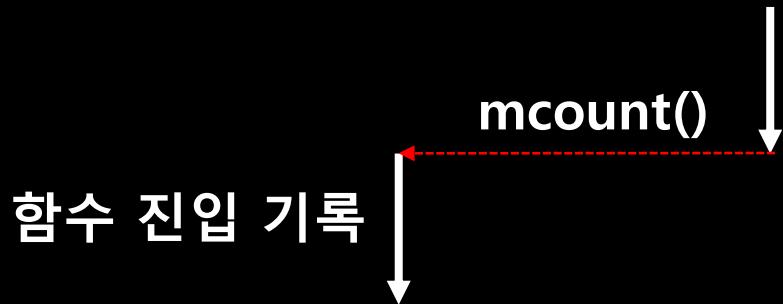
uftrace/libmcount/mcount.c

mcount_ret_stack

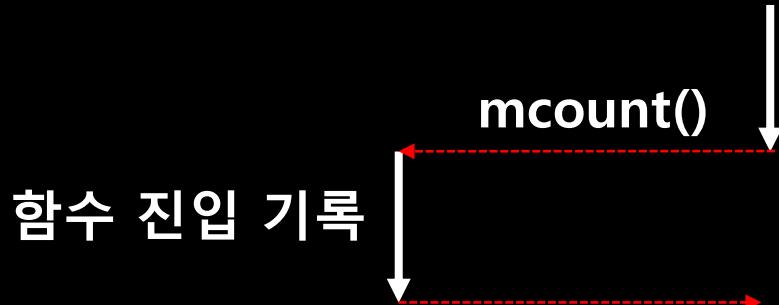
- **mcount_ret_stack** 은 함수 진입과 반환시에 다양한 정보를 기록하며 관리

```
struct mcount_ret_stack {
    unsigned long *parent_loc; /* 실제 return 주소값의 주소 */
    unsigned long parent_ip;   /* 실제 return 주소 */
    unsigned long child_ip;   /* mcount 다음 주소 */
    enum mcount_rstack_flag flags;
    /* time in nsec (CLOCK_MONOTONIC) */
    uint64_t start_time;     /* 함수 진입(entry) 시간 */
    uint64_t end_time;       /* 함수 반환(exit) 시간 */
    int tid;
    ...
    struct plthook_data *pd;
    /* set arg_spec at entry and use it at exit */
    struct list_head *pargs;
};
```

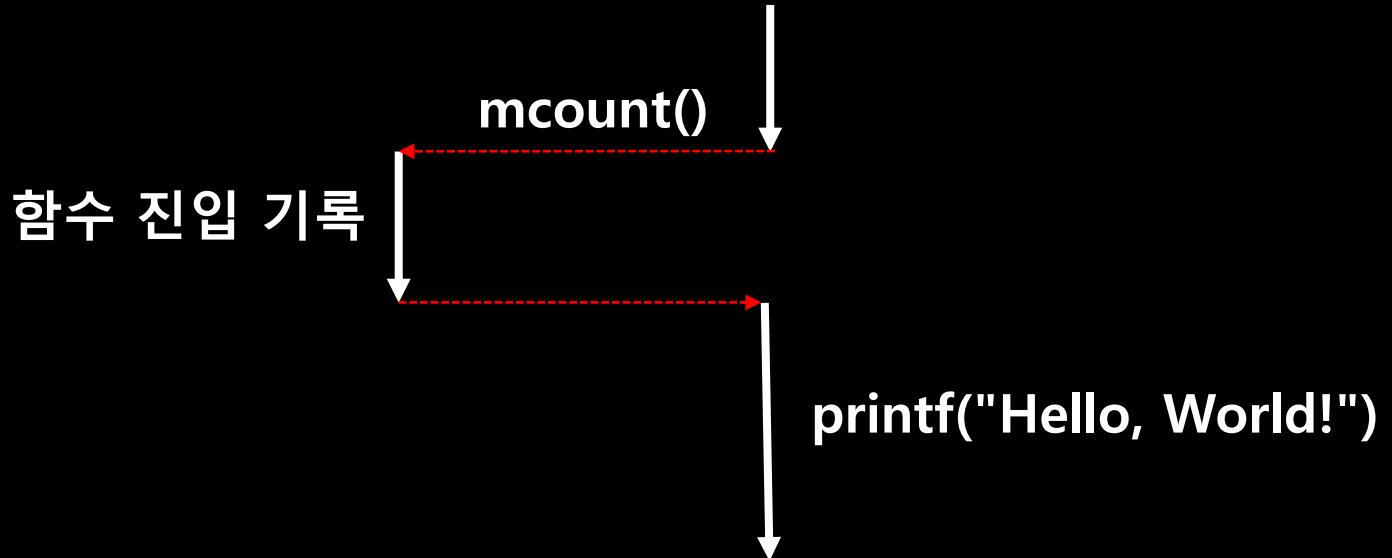
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



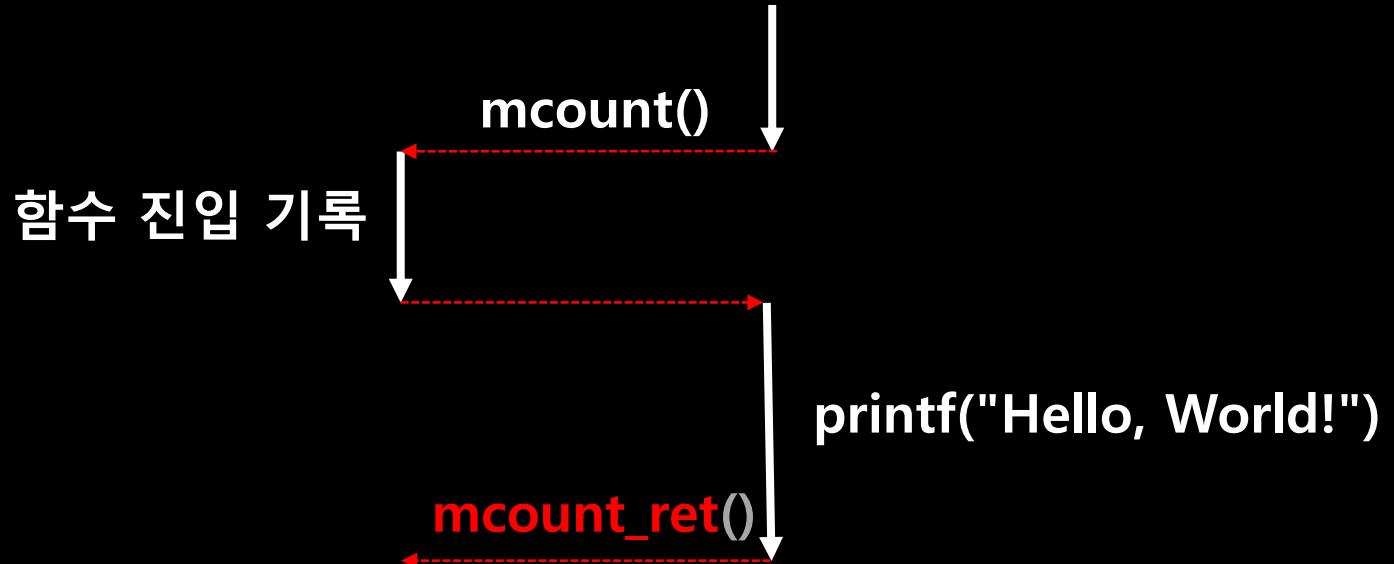
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



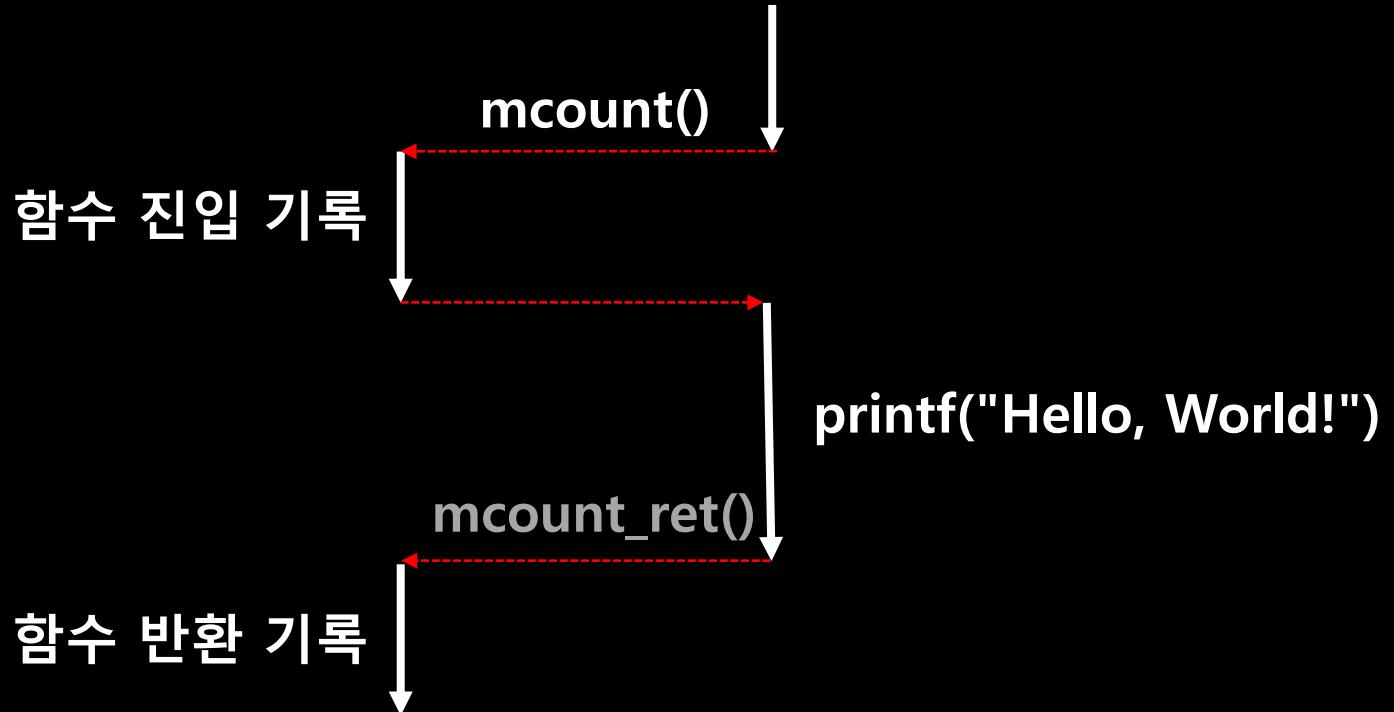
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



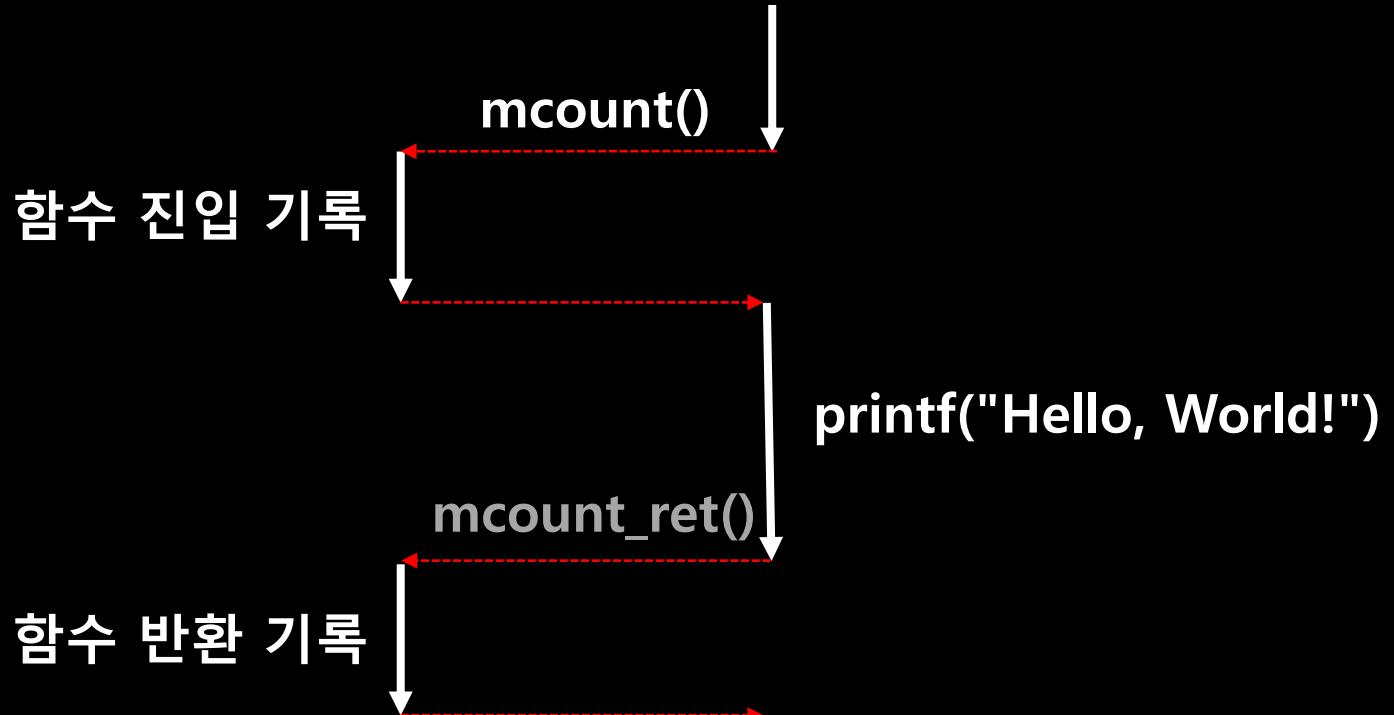
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



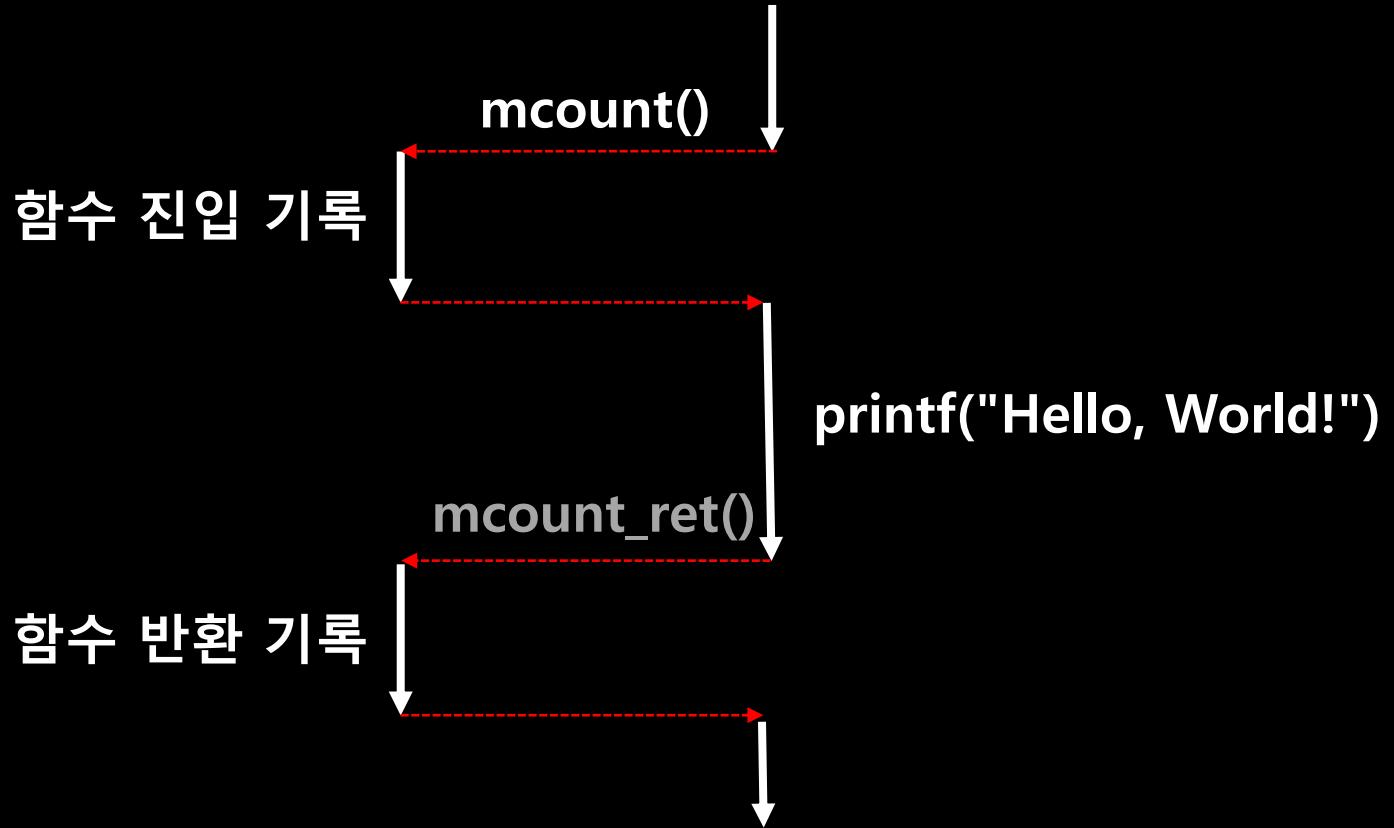
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



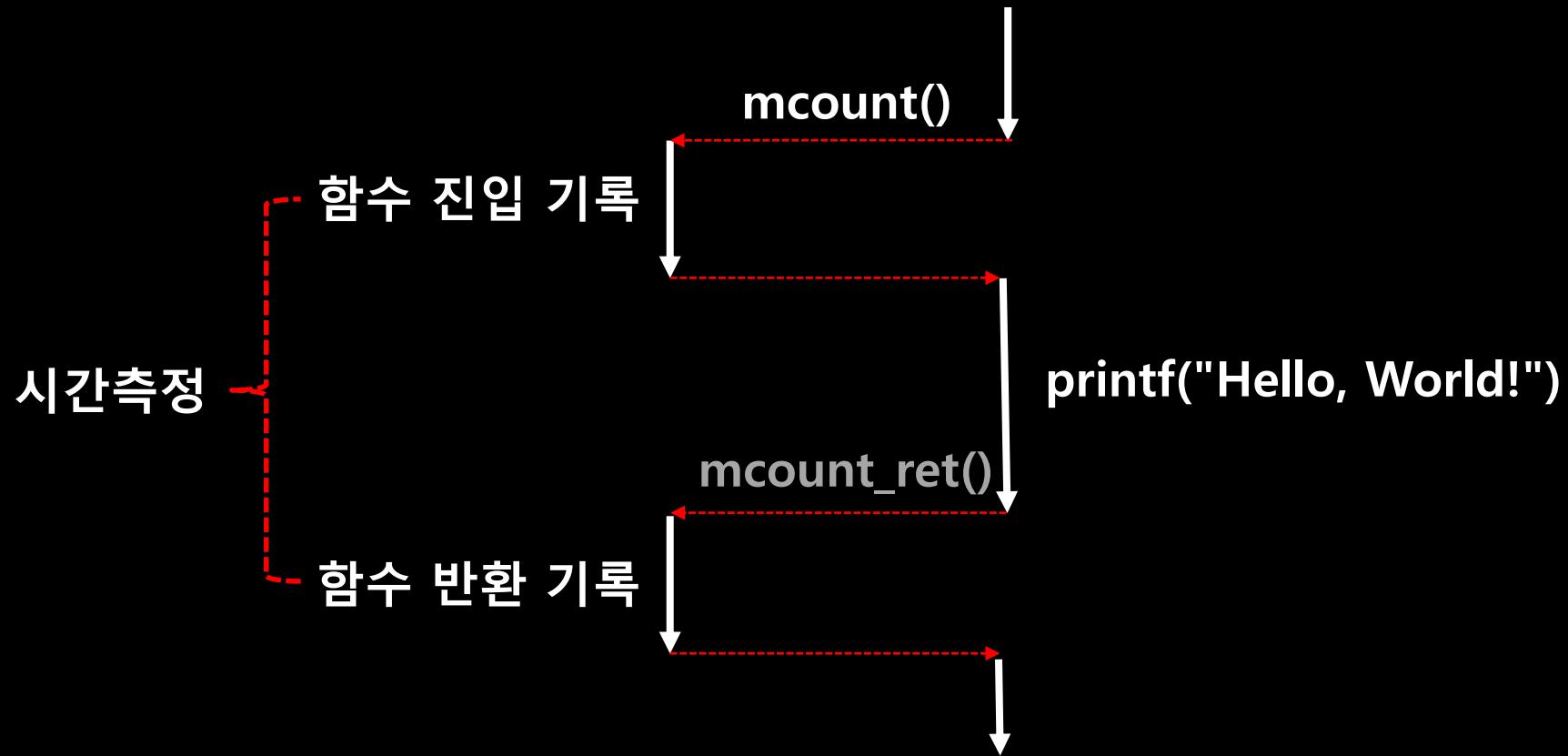
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



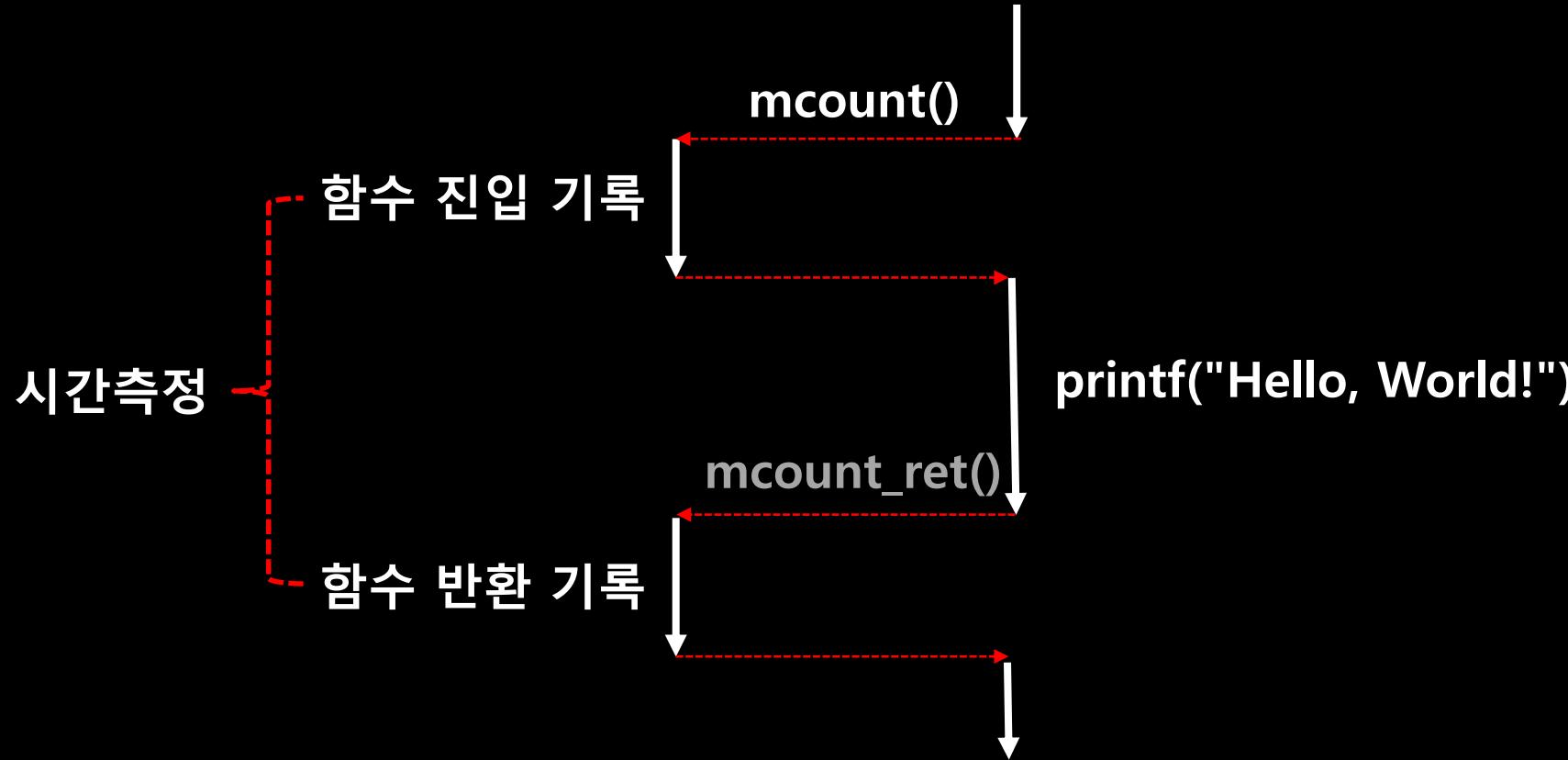
```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



```
$ uftrace ./a.out  
LD_PRELOAD=libmcount.so      a.out
```



함수 진입 및 반환 기록을 위해 16바이트씩 shared memory에 저장

Full Dynamic Tracing

Tracing without compiler assist

<https://uftrace.github.io/slides/#full-dynamic-tracing>

향후 활용 가능성

- **uftrace** 의 데이터는 **portable** 하게 설계됨
 - record 된 결과인 **uftrace.data** 디렉토리는 다른 환경 어디에서도 replay 하면서 성능 분석 가능
 - 예) ARM 환경에서 record 된 데이터를 x86_64 에서 분석
- 활용 방안
 - 다른 오픈소스 개발자들의 진입 장벽 완화
 - GCC, LLVM, node.js, chromium 등
 - GDB 와 같이 널리 사용되는 디버깅 툴
 - 처음부터 crash 가 발생한 지점까지 record 된 결과 공유
 - OS, 시스템 프로그래밍 교육을 위한 툴

감사합니다

<https://github.com/namhyung/uftrace>

<https://gitter.im/uftrace/ko>

Appendix

uftrace 소스 트리 구조

```
$ tree -d uftrace
uftrace
└── arch      # architecture 특화된 코드를 관리
    ├── aarch64
    ├── arm
    ├── i386
    └── x86_64
── check-deps # configure 스크립트 실행시 기능 테스트
── cmds       # 명령어(record, replay 등) 코드 관리
── doc        # 문서 관련 코드 (man page, slide 문서)
── gdb        # gdb 디버깅을 위한 편의 스크립트
    └── uftrace
── libmcount  # libmcount.so로 컴파일되어 대상 프로그램과 함께 동작
── libtraceevent # 커널 함수 정보가 기록된 버퍼를 읽는 코드
    └── include
        ├── asm
        └── linux
── misc       # 기타 등등...
── scripts    # -S 옵션이나 script 명령어로 실행할 예제 스크립트
── tests      # 테스트 관련 코드들
    └── arch
        ├── arm
        └── x86_64
└── utils     # libmcount.so와 uftrace에서 공통으로 사용하는 코드
```

```
$ tree -d uftrace
uftrace
└── arch      # architecture 특화된 코드를 관리
    ├── aarch64
    ├── arm
    ├── i386
    └── x86_64
── check-deps # configure 스크립트 실행시 기능 테스트
── cmds       # 명령어 (record, replay 등) 코드 관리
── doc        # 문서 관련 코드 (man page, slide 문서)
── gdb        # gdb 디버깅을 위한 편의 스크립트
    └── uftrace
── libmcount  # libmcount.so로 컴파일되어 대상 프로그램과 함께 동작
── libtraceevent # 커널 함수 정보가 기록된 버퍼를 읽는 코드
    └── include
        ├── asm
        └── linux
── misc       # 기타 등등...
── scripts    # -S 옵션이나 script 명령어로 실행할 예제 스크립트
── tests      # 테스트 관련 코드들
    └── arch
        ├── arm
        └── x86_64
└── utils     # libmcount.so와 uftrace에서 공통으로 사용하는 코드
```

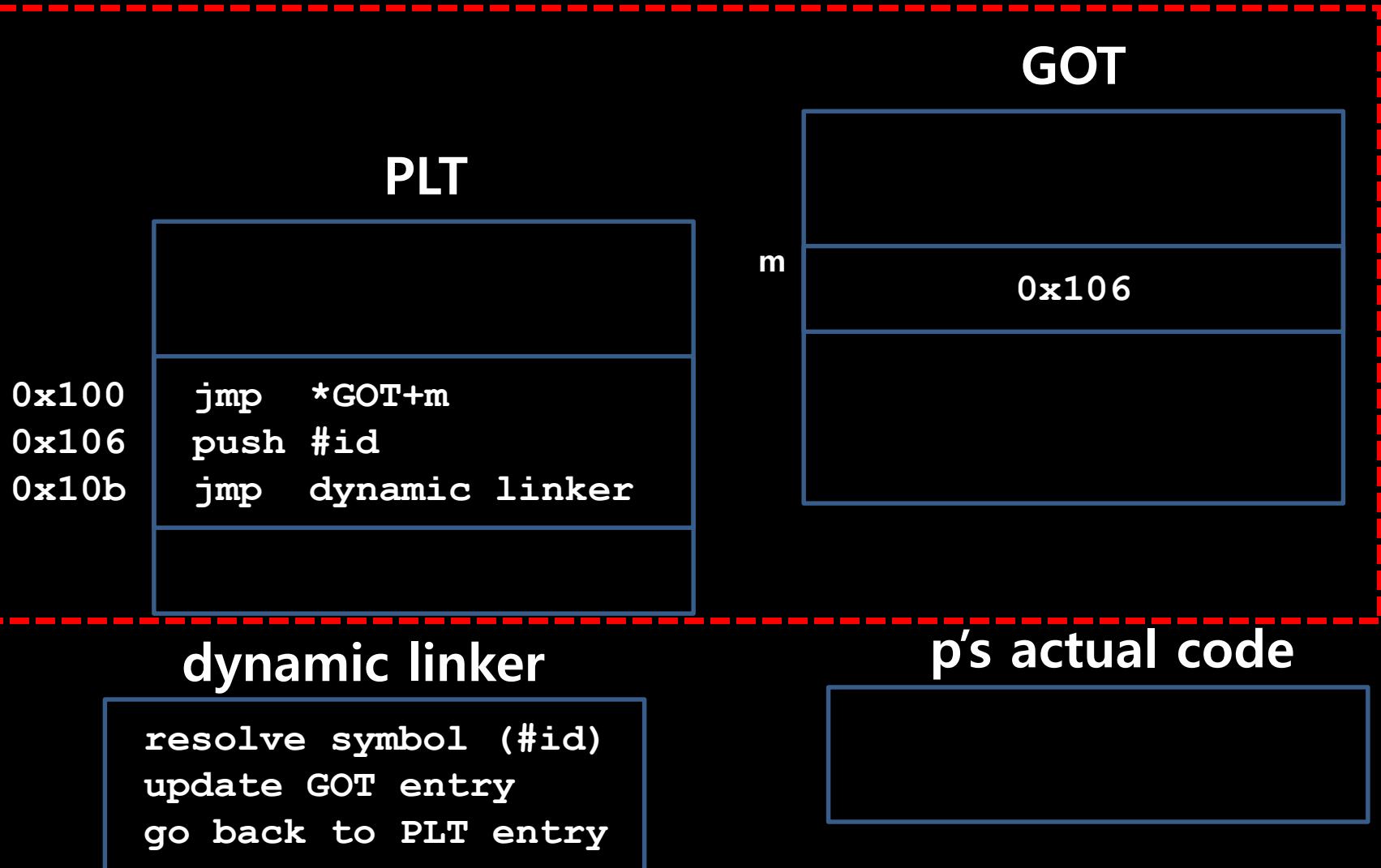
```
$ tree -d uftrace
uftrace
└── arch      # architecture 특화된 코드를 관리
    ├── aarch64
    ├── arm
    ├── i386
    └── x86_64
── cmds      # 명령어(record, replay 등) 코드 관리
── doc       # 문서 관련 코드 (man page, slide 문서)
── libmcount # libmcount.so로 컴파일되어 대상 프로그램과 함께 동작
── misc      # 기타 등등...
── scripts   # -S 옵션이나 script 명령어로 실행할 예제 스크립트
── tests     # 테스트 관련 코드들
└── utils    # libmcount.so와 uftrace에서 공통으로 사용하는 코드
```

Appendix

Library Hooking via PLT and GOT

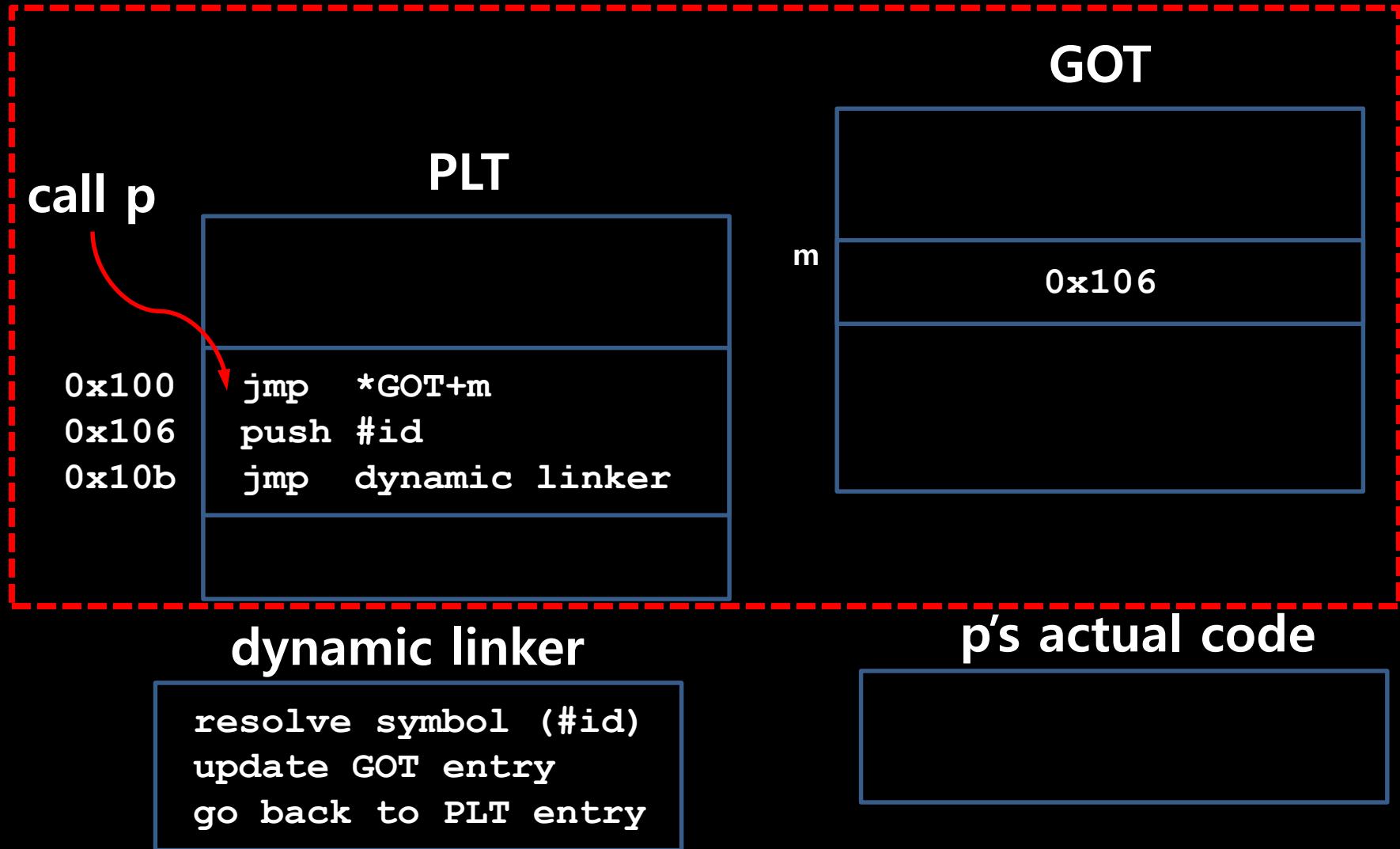
Dynamic Linking

- First Reference



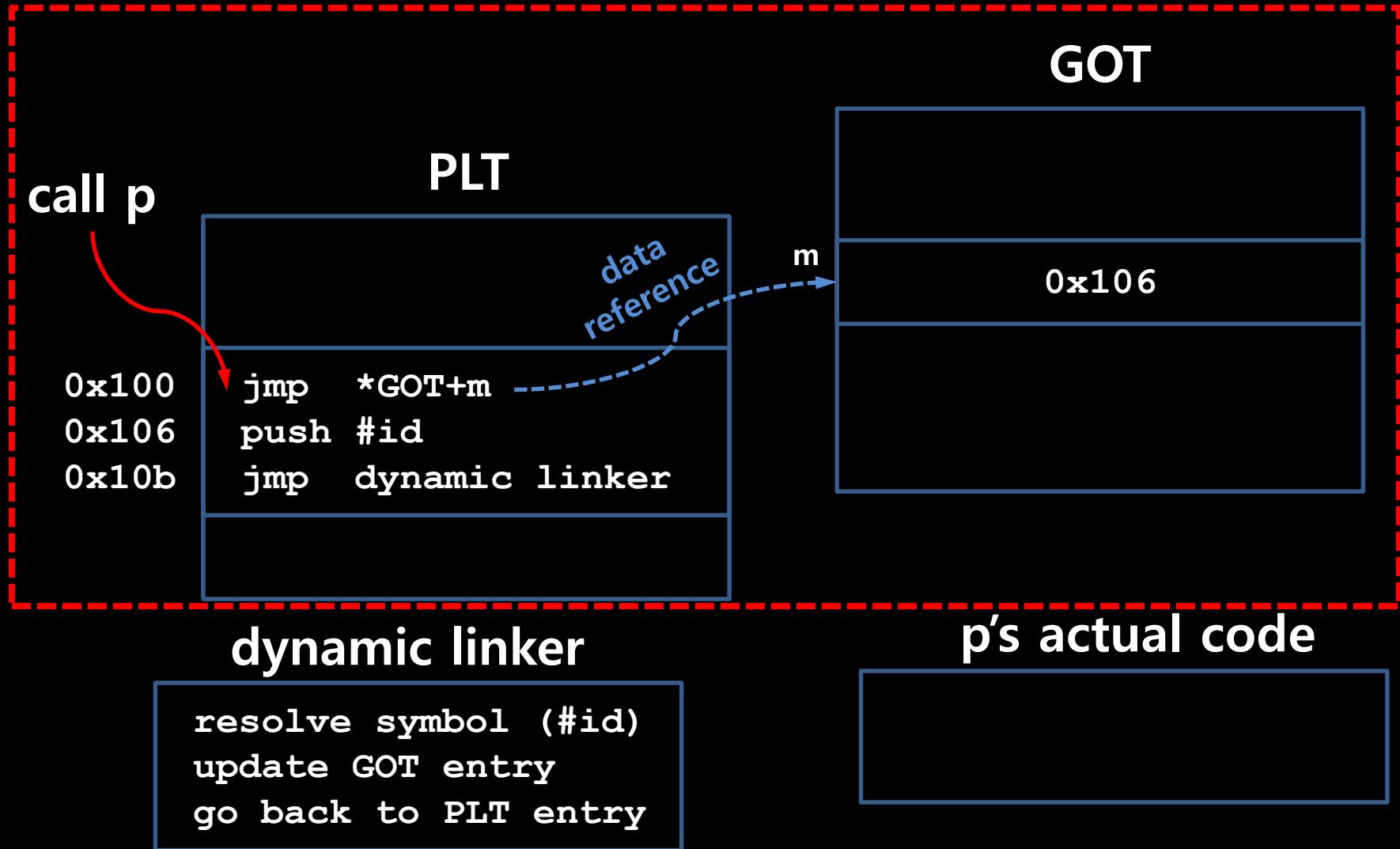
Dynamic Linking

- First Reference



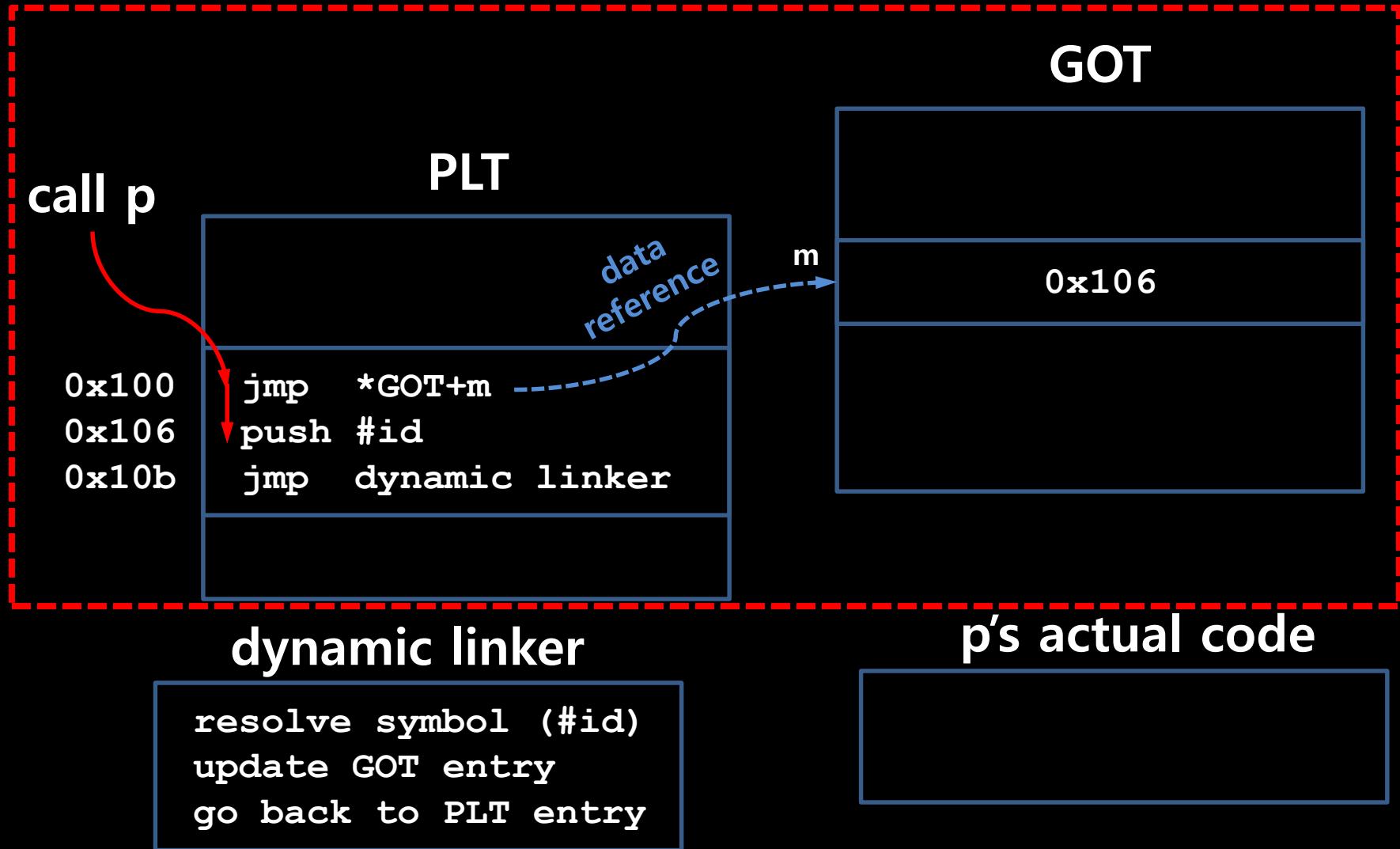
Dynamic Linking

- First Reference



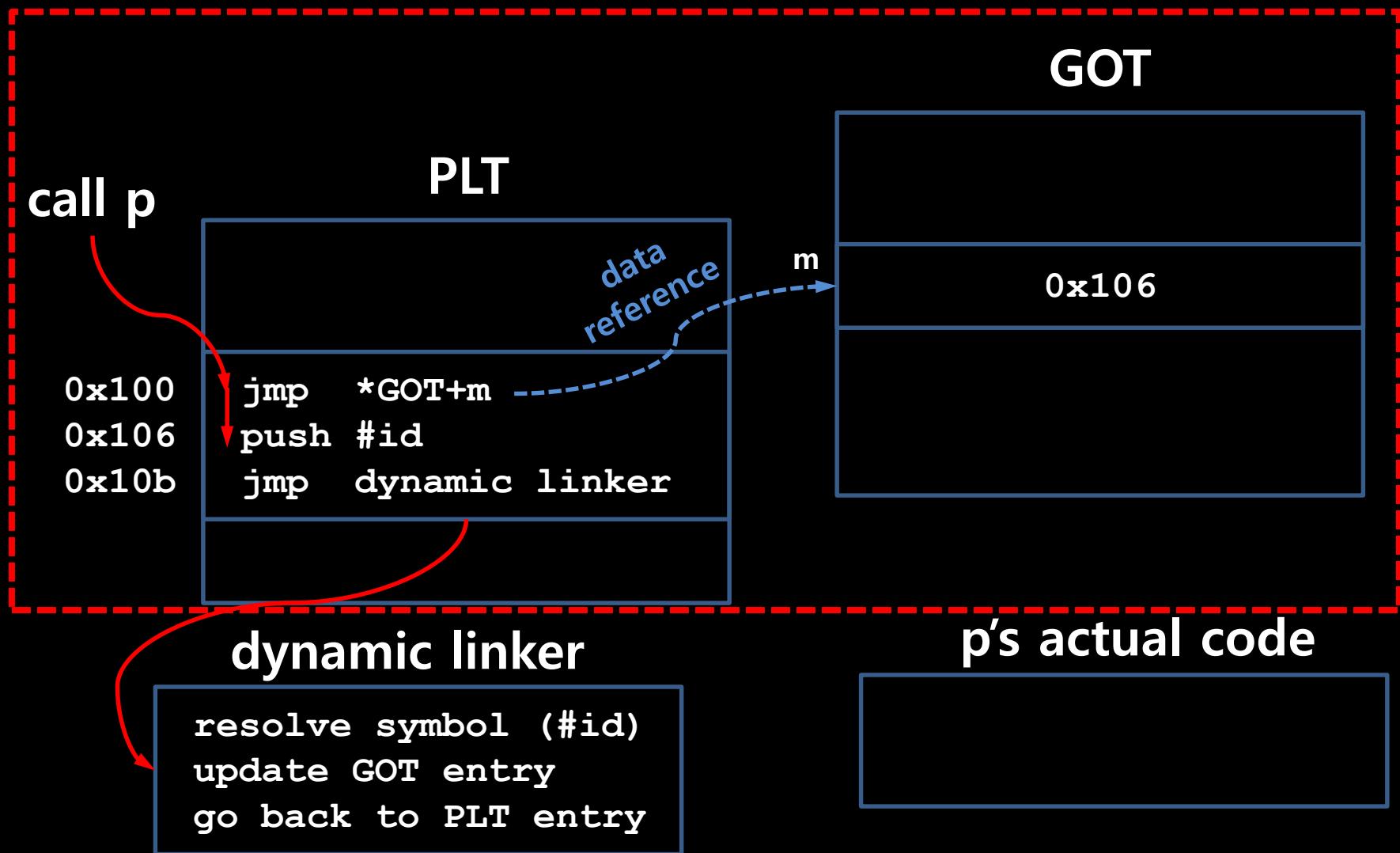
Dynamic Linking

- First Reference



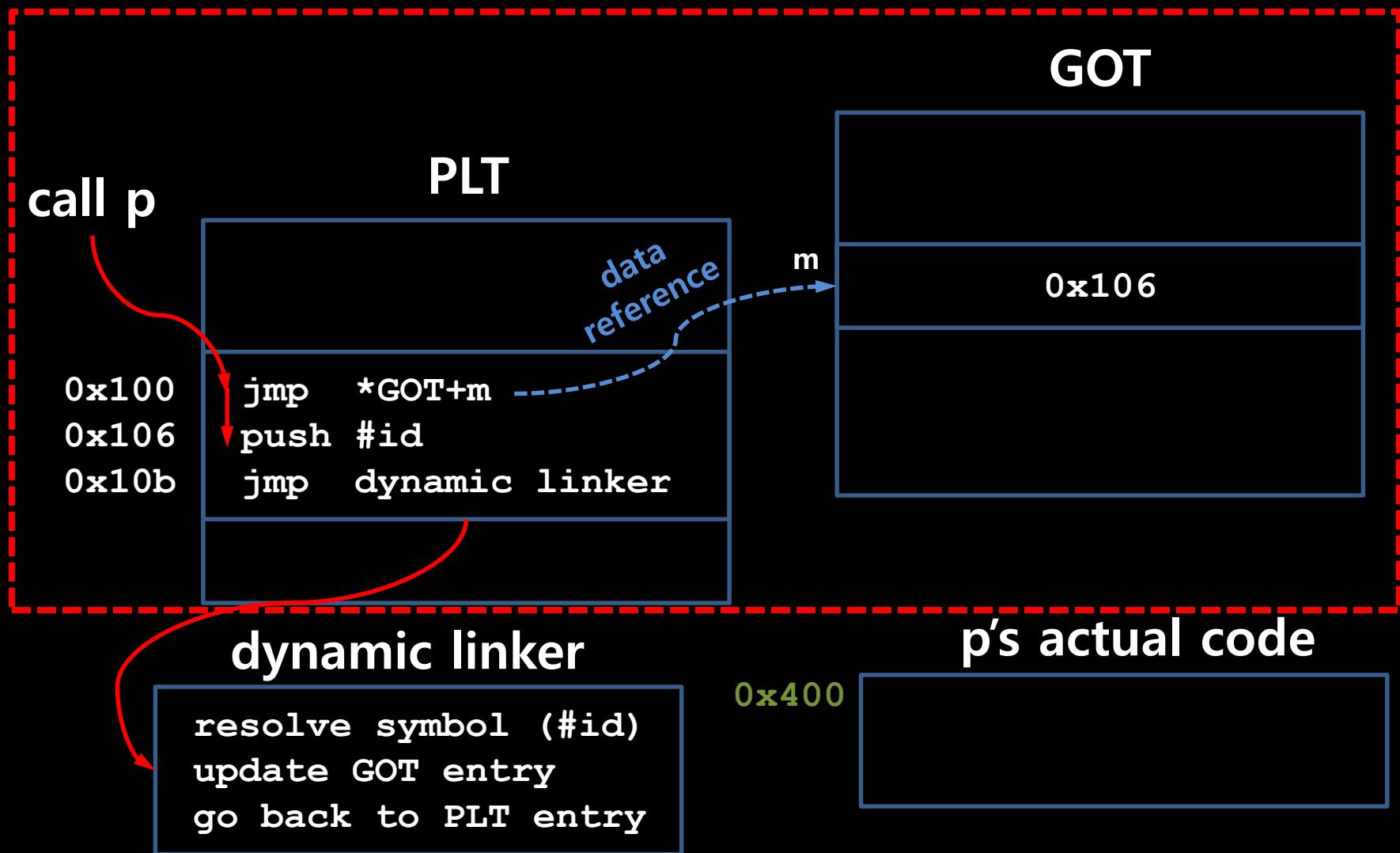
Dynamic Linking

- First Reference



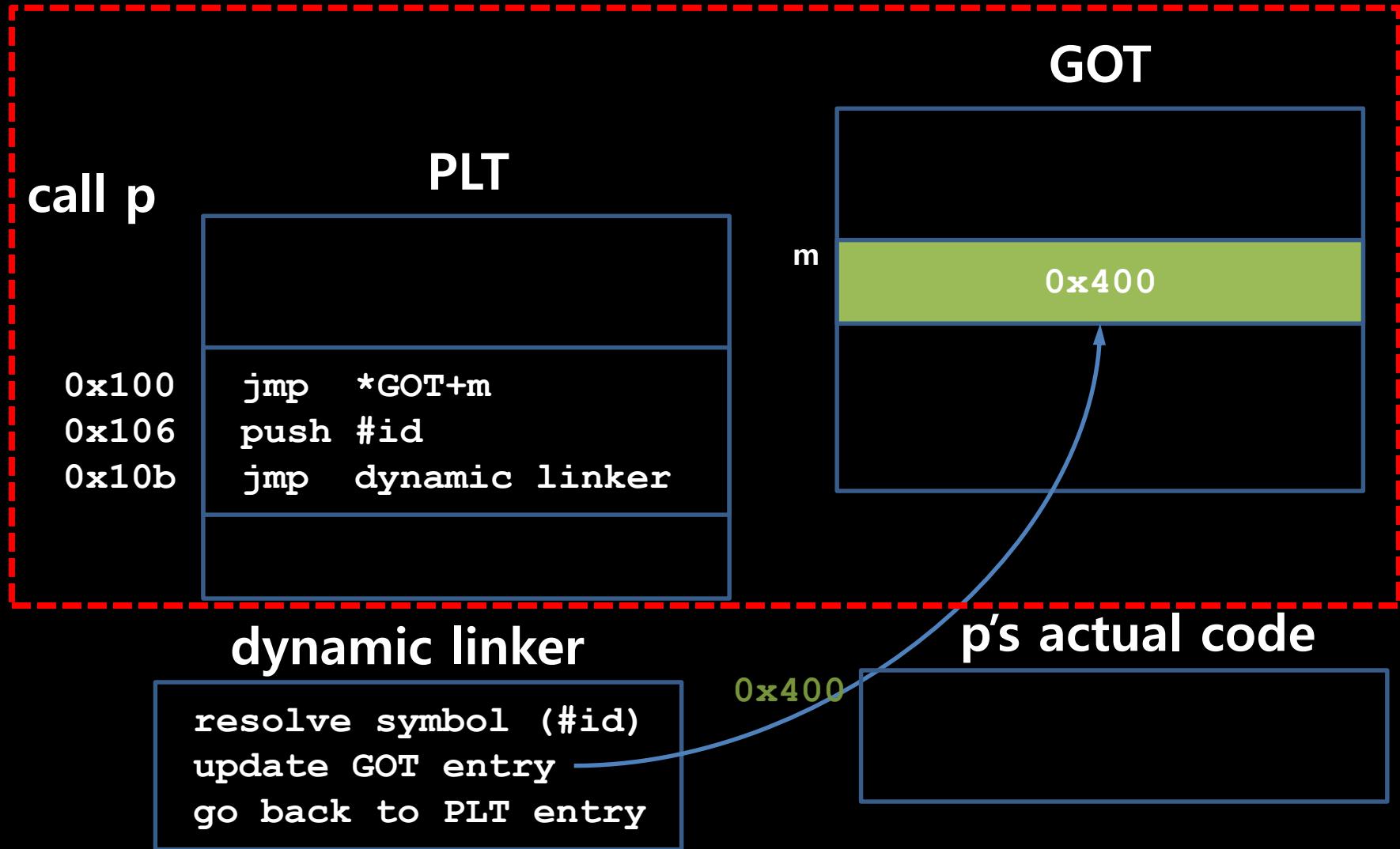
Dynamic Linking

- First Reference



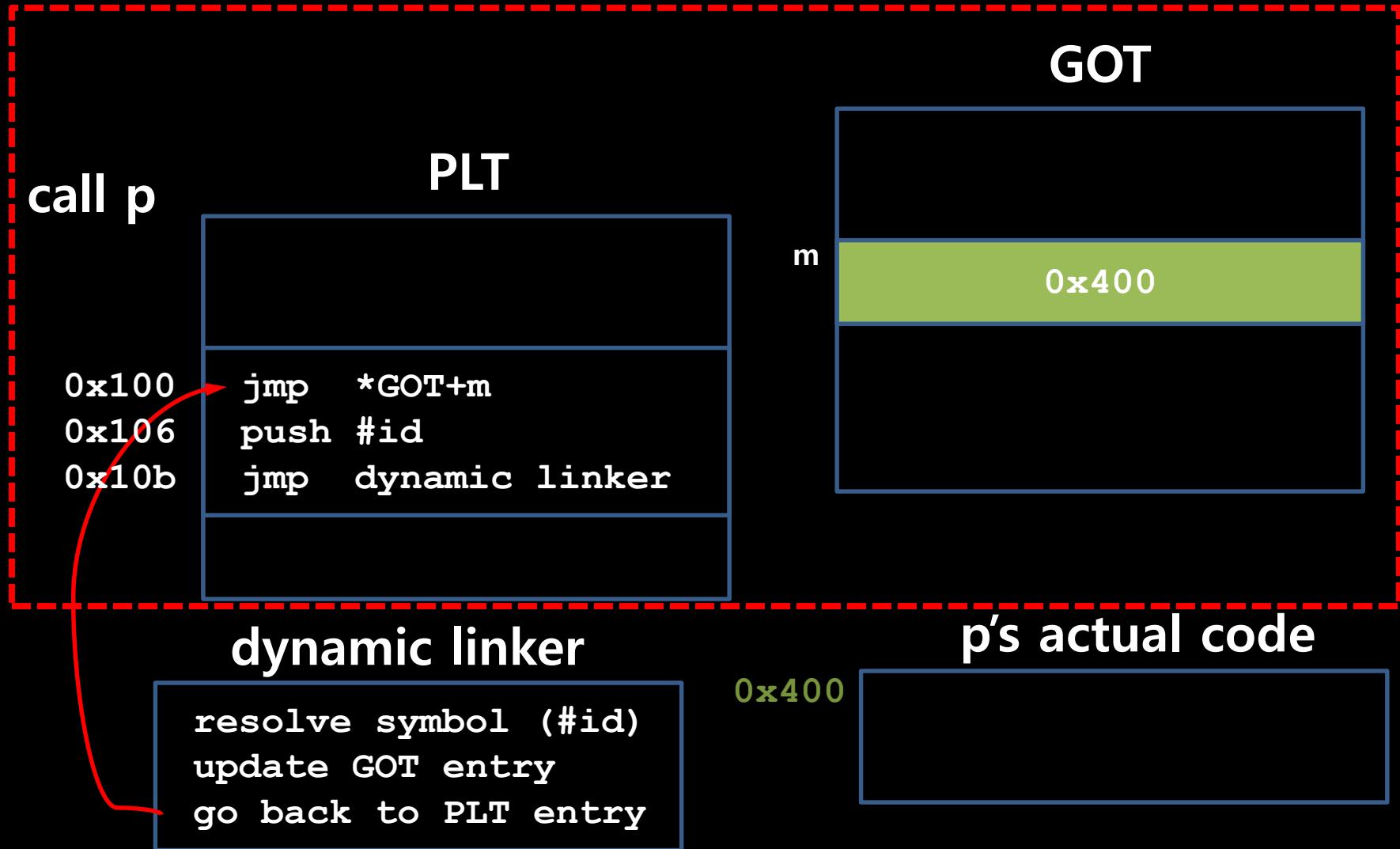
Dynamic Linking

- First Reference



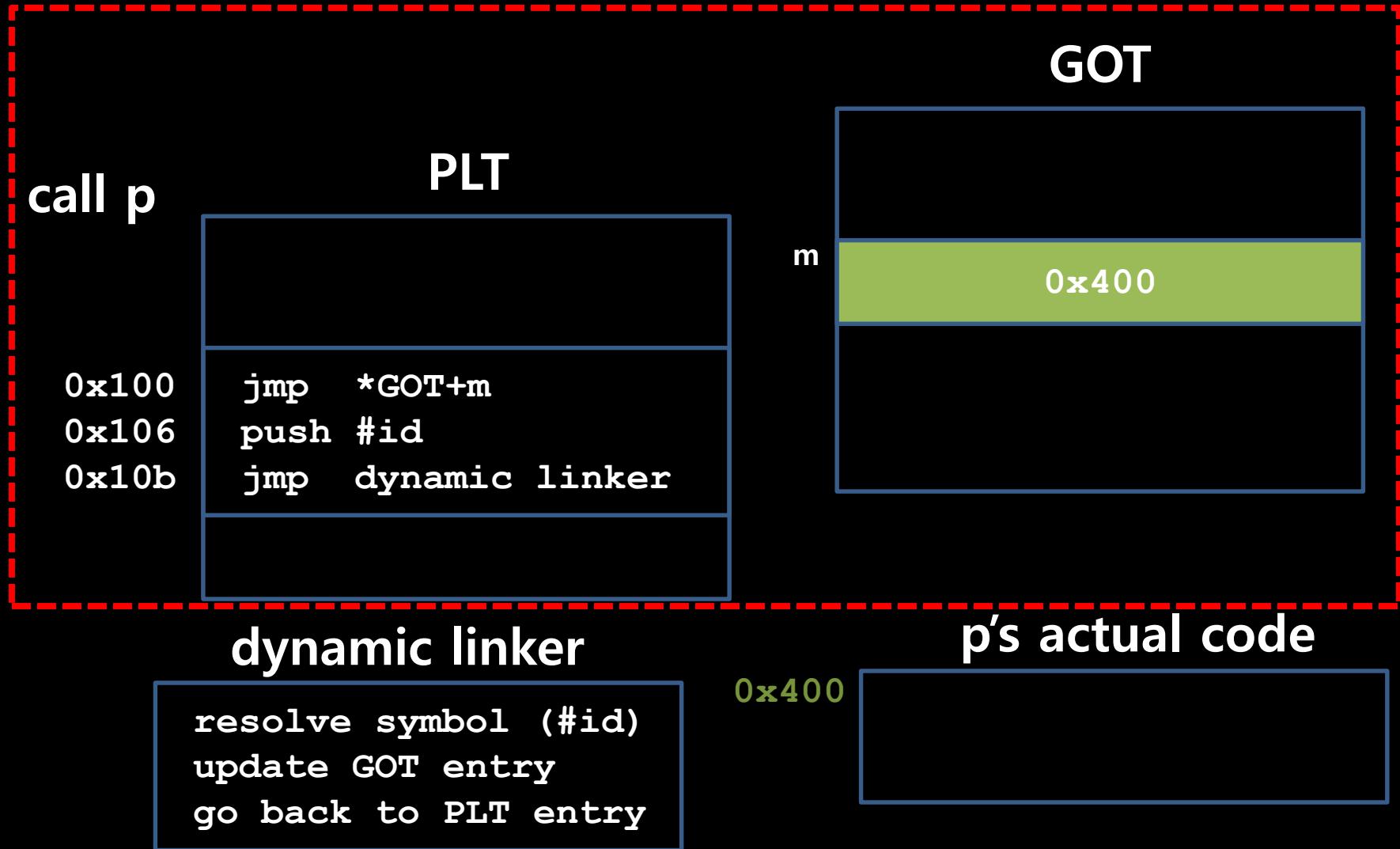
Dynamic Linking

- First Reference



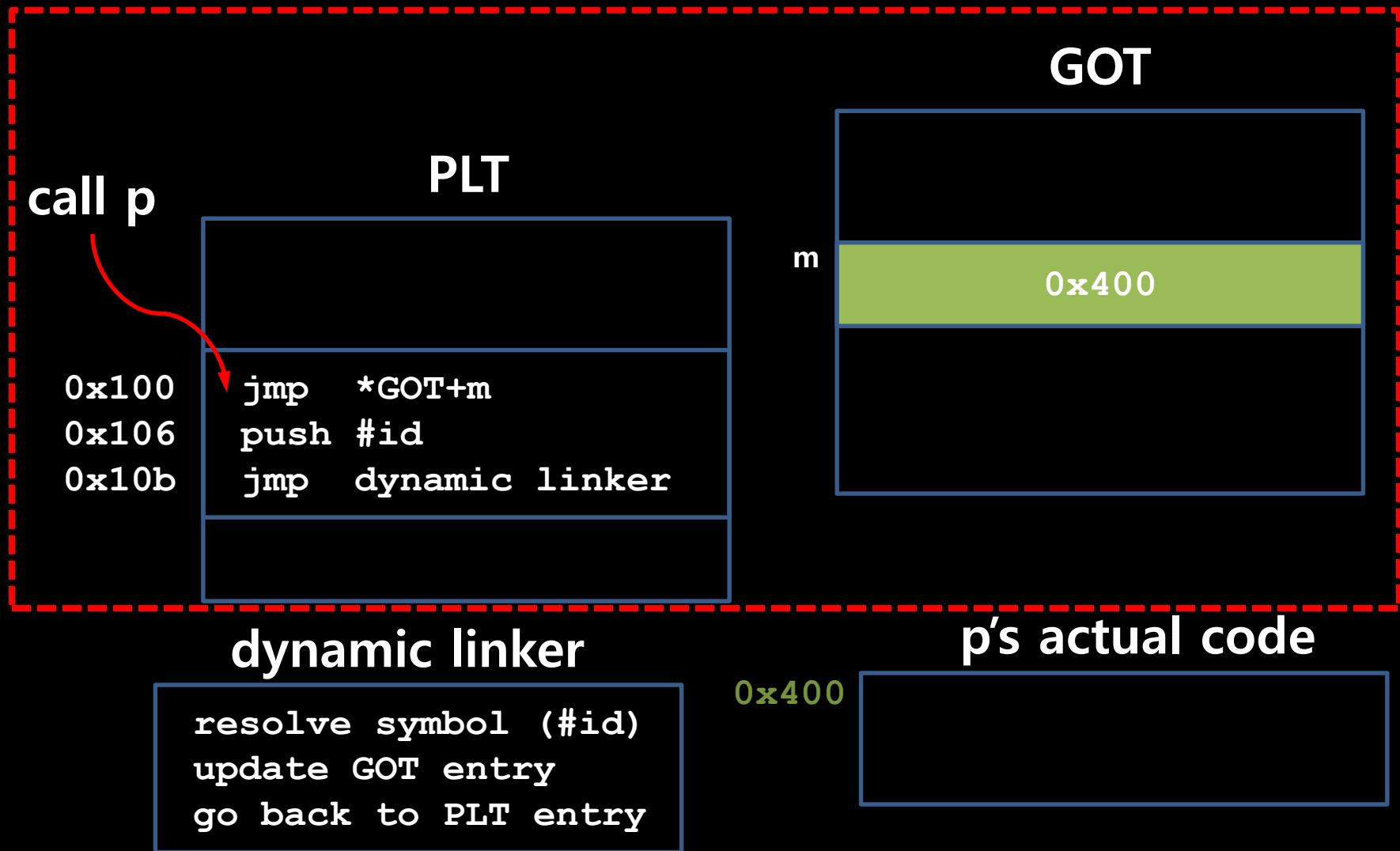
Dynamic Linking

- Later Reference



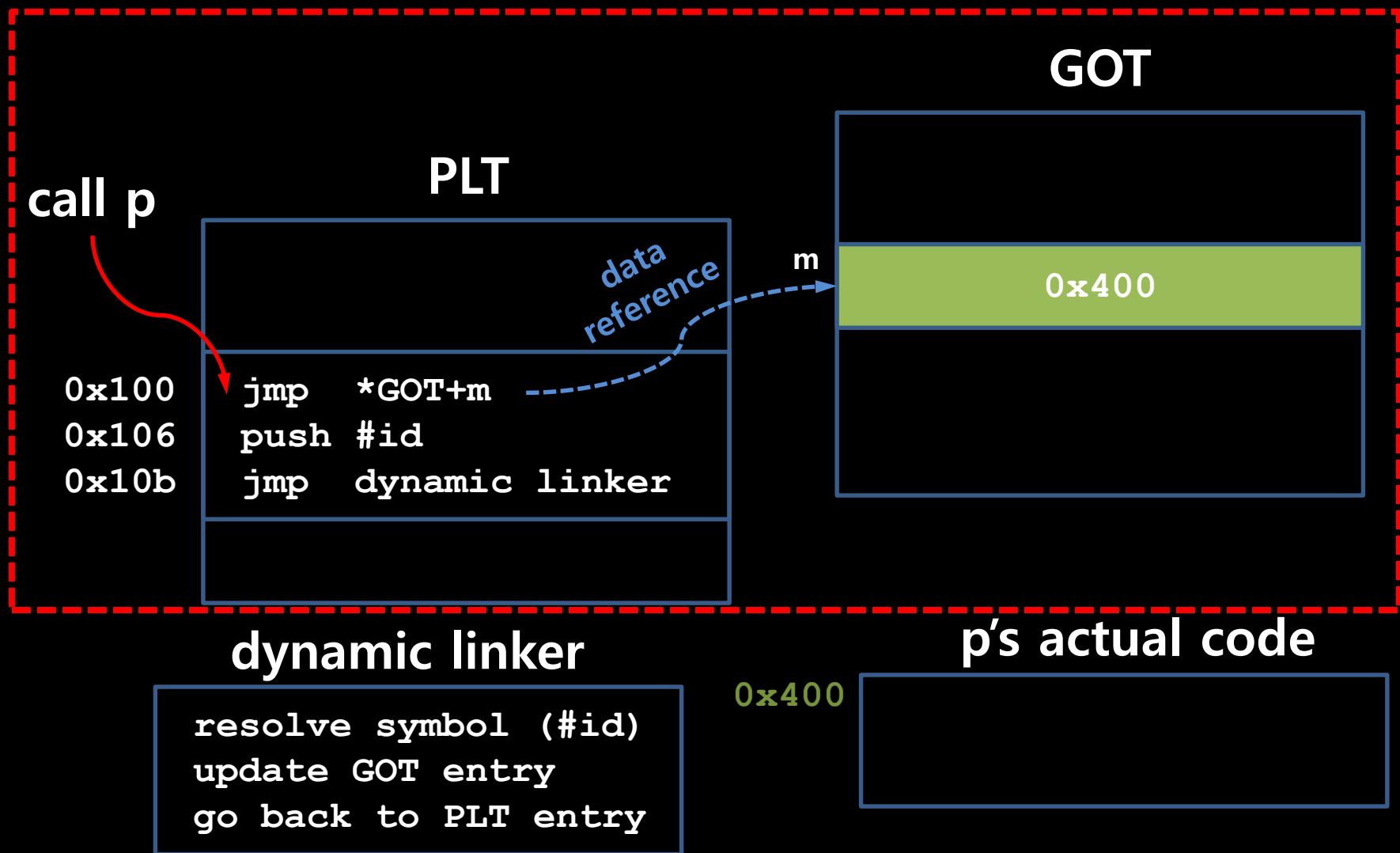
Dynamic Linking

- Later Reference



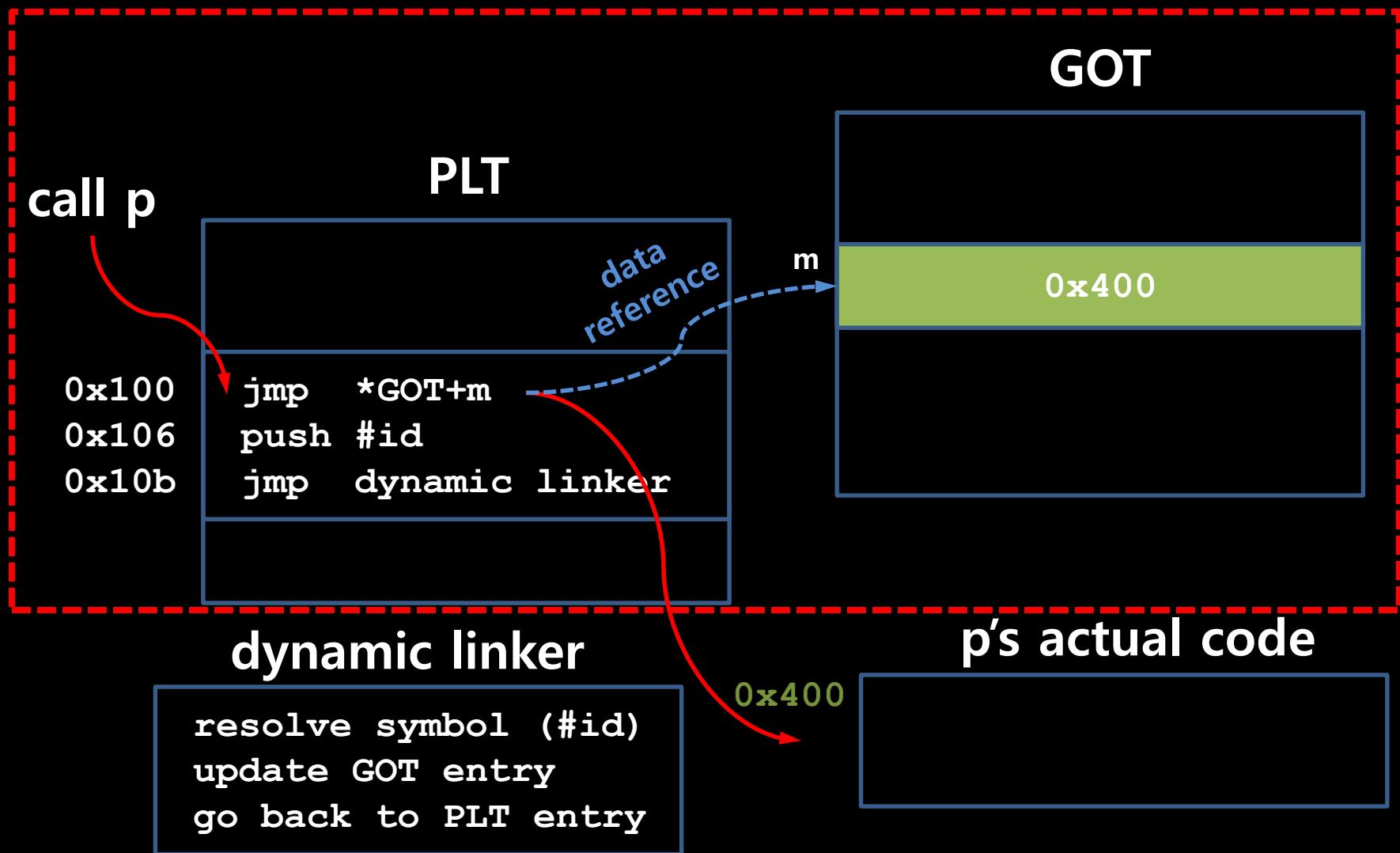
Dynamic Linking

- Later Reference



Dynamic Linking

- Later Reference



```

/*
 * mcount_arch_plthook_addr() returns the address of GOT entry.
 * The initial value for each GOT entry redirects the execution to
 * the runtime resolver. (_dl_runtime_resolve in ld-linux.so)
 *
 * The GOT entry is updated by the runtime resolver to the resolved address of
 * the target library function for later reference.
 *
 * However, utrace gets this address to update it back to the initial value.
 * Even if the GOT entry is resolved by runtime resolver, utrace restores the
 * address back to the initial value to watch library function calls.
 *
 * Before doing this work, GOT[2] is updated from the address of runtime
 * resolver(_dl_runtime_resolve) to utrace hooking routine(plt_hooker) .
 *
 * This address depends on the PLT structure of each architecture so this
 * function is implemented differently for each architecture.
 */
__weak unsigned long mcount_arch_plthook_addr(struct plthook_data *pd, int idx)
{
    struct sym *sym;

    sym = &pd->dsymtab.sym[idx];
    return sym->addr + ARCH_PLTHOOK_ADDR_OFFSET;
}

```

<https://github.com/namhyung/utrace/commit/4b283f7430d7c817cdf6d2999dc5a1608293c544>

```
static int find_got(struct uftrace_elf_data *elf,
                    struct uftrace_elf_iter *iter,
                    const char *modname,
                    unsigned long offset)
{
    bool plt_found = false;
    unsigned long pltgot_addr = 0;
    unsigned long plt_addr = 0;
    struct plthook_data *pd;
    ...
    pd = xmalloc(sizeof(*pd));
    pd->mod_name = xstrdup(modname);
    pd->pltgot_ptr = (void *)pltgot_addr;
    pd->module_id = pd->pltgot_ptr[1];
    pd->base_addr = offset;
    pd->plt_addr = plt_addr;
    ...
    pd->resolved_addr = xcalloc(pd->dsymtab.nr_sym, sizeof(long));
    pd->special_funcs = NULL;
    pd->nr_special = 0;

    mcount_arch_plthook_setup(pd, elf);
    list_add_tail(&pd->list, &plthook_modules);
    ...
overwrite_pltgot(pd, 2, plt_hooker);
    ...
    return 0;
}
```

uftrace/libmcount/plthook.c

```
static unsigned long __plthook_entry(unsigned long *ret_addr,
                                    unsigned long child_idx,
                                    unsigned long module_id,
                                    struct mcount_regs *regs)
{
    ...
    rstack = &mtdp->rstack[mtdp->idx++];
    rstack->depth      = mtdp->record_idx;
    rstack->pd         = pd;
    rstack->dyn_idx    = child_idx;
    rstack->parent_loc = ret_addr;
    rstack->parent_ip   = *ret_addr;
    rstack->child_ip    = sym->addr;
    rstack->start_time  = skip ? 0 : mcount_gettime();
    rstack->end_time    = 0;
    rstack->flags       = skip ? MCOUNT_FL_NORECORD : 0;
    rstack->nr_events   = 0;
    rstack->event_idx   = ARGBUF_SIZE;

    /* hijack the return address of child */
*ret_addr = (unsigned long)plthook_return;

    /* restore return address of parent */
    if (mcount_auto_recover)
        mcount_auto_restore(mtdp);
}
```

uftrace/libmcount/plthook.c